Motion Dynamics Animation Workshop

Siggraph 2001, Course #23

Course Presenter
Michael O'Rourke

Pratt Institute
Brooklyn, NY 11205

Special Note to all Course participants:

Please bring your copy of these Course Notes with you to the Workshop; you will need to follow the Exercise portions of these Notes as you work at the machines. The Course Notes will be available on line at each workstation, but you may prefer to work from these hardcopy Notes.

Copyright Notice

Acknowledgements

I wish to thank the Siggraph organization and in particular, Lou Harrison, Courses Chair; Damon Hill, Student Volunteers Chair; Stuart Anderson, Creative Applications Lab Chair; and Lynn Pocock, Siggraph 2001 Chair, for their help in organizing this Workshop. I also want to express my gratitude to Alias|Wavefront for generously contributing their support to the Workshop. Screen-snaps of the Maya interface are used with their permission.

I also want to thank all of my colleagues at Pratt Institute for their support, especially Beth Warshafsky, Chair of the CGIM Department, and Bill Burg, my Administrative Assistant for this Course.

And very importantly, I want to thank all of my graduate students at Pratt who served as Animation Assistants to this Workshop.

Contents

Author Biography

Michael O'Rourke is an artist and animator and Professor in the Department of Computer Graphics and Interactive Media at Pratt Institute, in Brooklyn, NY. His professional training in the arts includes an M.F.A. degree from the University of Pennsylvania. Following his studies, he was a Senior Research Staff Artist at the New York Institute of Technology Computer Graphics Laboratory, where he worked on personal artwork and animations, as well as commercial animations, contributing to a Clio-award winning animation and a first-prize-winner at the Los Angeles Animation Celebration.

At Pratt, he is the senior faculty member in the animation program of the Department of Computer Graphics and Interactive Media. He is also the author of *Principles of Three-Dimensional Computer Animation* (W.W. Norton, 1998, Revised Edition). This book has been published in Japanese (Toppan Co., Tokyo, Japan), in Chinese (MegaViz, Taipei, R.O.C.), and in Korean (Doo Nam Co., Seoul, Korea). He is an active artist, concentrating most recently on large-scale interactive mural installations. He has exhibited his work internationally, including exhibitions at Siggraph 2001, and a solo exhibition of prints at the Hong Gah Museum, Taipei, Taiwan. He has also done several series of computer-aided sculpture and graphic works for the artist Frank Stella.

In addition to his experience as an artist, he has broad experience as an educator. His studies in this area were at Harvard University, where he earned an Ed.M. degree. In addition to his teaching at Pratt, he has taught Kindergarten, English as a foreign language in West Africa, and conversational French.

Contact Information:

Michael O'Rourke
Pratt Institute
Dept. of Computer Graphics & Interactive Media
200 Willoughby Ave.
Brooklyn, NY 11205

mor@michaelorourke.com
www.michaelorourke.com

Introduction

This course is an intermediate level, hands-on workshop designed to introduce participants to the principles and practice of motion dynamics animation. This is approached through a combination of lecture presentations and hands-on exercises using one of today's major high-end 3D software packages, *Maya 3.0*. Both rigid-body and soft-body dynamics are introduced.

Given the brevity of the course, there is no expectation that participants will become expert in the techniques presented. Instead, the hope is that they will come to understand the core principles presented and begin to see the possibilities of their implementation through the tutorial exercises. These exercises will remain on the workstations of the Creative Applications Laboratory throughout the week of the conference. Those with an interest in improving their understanding and skill beyond what is possible in this course can practice as much as they want during the week.

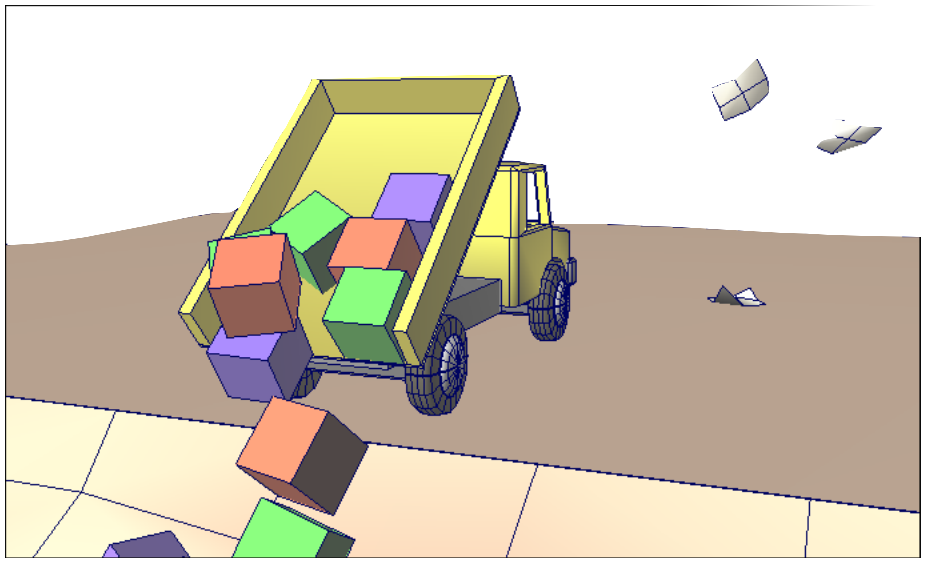
The course is divided into two parts, with each part consisting of a lecture presentation on the principles of motion dynamics; a demonstration of how these principles are implemented on the *Maya* software package; and a series of command-by-command tutorial exercises in which participants use *Maya* to develop short animations utilizing those principles.

The written Notes for this course, printed here, follow the structure of the course itself—that is, two parts, with each part consisting of a brief lecture followed by the tutorial exercises.

Lecture 1 of these Notes is reprinted, with permission, from the author's book, *Principles of Three-Dimensional Computer Animation* (W.W. Norton, 1998, Revised Edition. 288 pages, 332 illustrations. ISBN 0-393-73024-7. $55.00 USA).

Lecture 1

Rigid-Body Motion Dynamics



All material in Lecture 1, both text and lecture, is taken from *Principles of Three-Dimensional Computer Animation* (W.W. Norton, 1998, 1995), by Michael O'Rourke. It is reprinted here by permission.

Rigid-Body Motion Dynamics

In certain situations the movement of an object, such as a ball falling from a height and bouncing on a surface, can be predicted very precisely. Physicists have studied this sort of phenomenon for a long time and have developed very accurate mathematical formulas to describe what happens in this situation. Knowing the effect of gravity and various properties of the ball and of the surface, they can calculate, using the laws of physics, how quickly the ball will fall, as well as how high and how often it will bounce.

This sort of mathematically precise and detailed study of the motion of objects is called **motion dynamics** and can be of great use to animators. The ball dropping straight down and bouncing off a flat surface is not very difficult to animate using standard keyframing techniques, but a more complex situation, such as a box being dropped and bouncing down a stairway (Figure 4-11), is very difficult to animate in a naturalistic way using keyframing. To estimate realistically how far the box bounces with each bounce, how much and in what directions the box rotates as it flies through the air, which edge or corner of the box strikes the surface of the stairs first, and so on, is a *very* complicated piece of animation. The physical laws of motion dynamics, however, are precise enough for you to calculate the motion of the box.

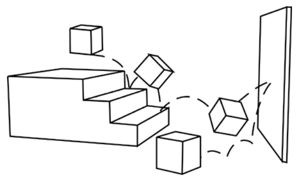


Figure 4-11. The motion of some objects can be calculated precisely according to the laws of physics.

The software of many sophisticated 3D animation packages incorporates this sort of motion-dynamics calculation. The best of these systems provide user interfaces that make the technical complexity of the underlying calculations invisible. Typically, you use a mouse device to open a menu and make several selections. These selections define the various parameters that control the motion of the object to be animated. Once you have defined these parameters, you make a selection that instructs the system to perform the calculations and play back the resulting animation on the screen. This result is called a **simulation**, because it simulates, or mimics, the behavior of objects in a real-world situation behaving under the influence of real-world physical laws.

In setting up a simulation you first define the **physical properties** of the object to be animated. That is, the motion-dynamics system must know certain characteristics of the object in order to calculate the simulated movement of the object. For example, a ball made of solid iron bounces much differently than does a ball made of balsa wood.

In physics—and in motion dynamics—one of the most basic physical properties of an object is **mass**. Mass is related to, but not quite the same as, weight. That is, the weight of an object is the result of gravity pulling down on the massiveness of the object. On Earth, gravity is effectively the same everywhere, so mass and weight have a direct, one-to-one correspondence. However, since weight is linked to gravity, the weight of an object *can* change—for example, if the object were in outer space—although the mass of the object remains the same wherever the object is. An object with a large mass is very difficult to start moving and very difficult to stop once it begins moving, and this property directly affects how the object moves in a motion-dynamic simulation. For example, if you roll a massive ball down a ramp and onto a level surface, the ball tends to keep rolling for a long time because the massiveness overcomes the forces that otherwise would make the object stop (Figure 4-12a). If you roll a less massive ball down the same ramp, it tends to stop much more quickly on the flat surface (Figure 4-12b).

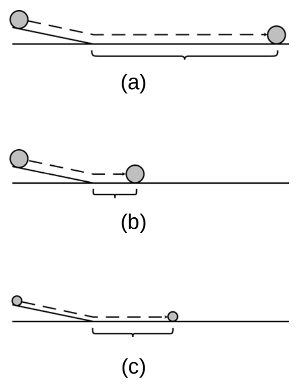


Figure 4-12. The mass of an object plays a key role in how the object behaves as it moves. Mass is a function of both the density of the material and the size of the object.

What makes the one ball more massive than the other is not the size of the objects but the **density** of the materials from which they are made. A balsa-wood ball is less massive than an iron ball because balsa wood is less dense than iron. A solid clay ball of the same size is more dense and more massive than a balsa-wood ball, but less dense and less massive than an iron ball.

Size does play a role, however, in determining mass, which is a function of both the density and the size (more exactly, the volume) of an object. A small iron ball (Figure 4-12c), even though it has the same density as a larger iron ball (Figure 4-12a), has a smaller mass because it has a smaller volume. Consequently, the little, less massive ball does not roll as far as the larger, more massive ball (Figure 4-12b).

Since the mass of an object depends on both volume and density, most people cannot readily estimate it. Consequently, many systems allow you to define mass indirectly by specifying density. By thinking in terms of material (iron, wood, clay, etc.) you can readily give the system a density parameter value—larger numbers for more dense materials, smaller numbers for less dense materials. The software then automatically calculates the volume of the object and then, from the combination of volume and density, the mass.

The density of an object also is important to consider when simulating objects lighter than air. A gas-filled balloon, for example, rises into the air because the gas inside the balloon is of a very, very low density. This causes the total mass of the balloon to be so little that gravity will not even keep it on the ground.

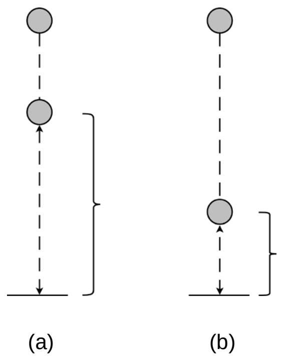


Figure 4-13. The elasticity of an object is a measure of how much the object bounces when it hits a surface.

Another important physical property of objects, called **elasticity**, reflects the way objects bounce. Imagine that two balls have exactly the same mass, but that when you drop these balls, one ball bounces much higher than the other one does (Figure 4-13a and b). A solid rubber ball, for example, tends to bounce much higher than a solid plastic ball—even though, as in this example, the masses of the balls happen to be exactly the same—and a glass marble dropped on a concrete floor bounces much higher than you might expect. Objects that have a high degree of elasticity, such as the solid rubber ball and the marble, tend to bounce a lot. Objects with lower elasticity bounce less.

The term "elasticity" can be a bit misleading, however. In day-to-day usage, the word describes objects easily stretched or deformed. A rubber band and a soft rubber ball that you can squeeze between your fingers are elastic in this sense. As it is used in motion dynamics, however, "elastic" means something different. A glass marble is extremely elastic because it bounces very high, even though it is very hard and does not deform between your fingers at all. In motion dynamics, elasticity refers to the amount of energy lost when an object makes contact with something else. If an object is very elastic, very little energy is lost and the object bounces a lot. If the object is very inelastic, a great deal of energy is lost and the object therefore bounces only slightly.

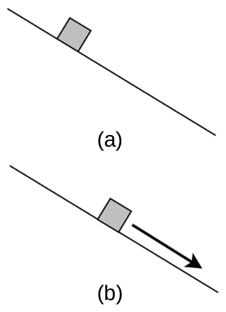


Figure 4-14. The static friction of an object is a measure of how readily the object, when stationary, will start to move after being subjected to some force.

Most systems also require you to define the **friction** created by an object. Since friction is a function of how smooth or rough a surface is, this property is sometimes referred to as **roughness**. The first type of friction, or roughness, is **static friction**, which prevents stationary objects from sliding down inclined planes. It is called "static" because it refers to nonmoving, or static, objects. If the static friction of a box is very great, for instance, the box will not slide at all (Figure 4-14a). If you decrease the static friction, the box will begin to slide (Figure 4-14b).

The second type of friction, **kinetic friction**, causes moving, or kinetic, objects to come to a stop. For instance, suppose you place two boxes identical in mass on two ramps identical in slope. What causes one box to stop sliding after only a short time is the kinetic friction, or roughness, of the surface of the box (Figure 4-15b). If the other box is made of some very slickly polished material and has very low kinetic friction, it will continue sliding for a longer distance (Figure 4-15a).

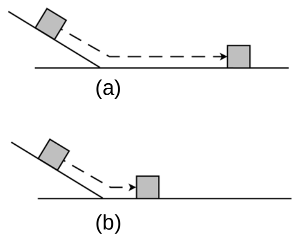


Figure 4-15. The kinetic friction of an object is a measure of how easily the object, when moving, can be stopped by some resisting force.

Static friction and kinetic friction function independently and therefore can be adjusted independently in most systems. For example, if you set the static friction of a ball rolling on a surface to zero, the ball will slide, rather than roll, along the surface. At the same time, you can adjust the kinetic friction of the ball up or down. If you increase the kinetic friction, the ball will slide (because the static friction is set to zero) and stop quickly (because the kinetic friction is high). If you decrease the kinetic friction, the ball will slide (because the static friction is still zero) and come to a stop slowly (because the kinetic friction is low).

So far, we have discussed a number of physical properties important for a motion-dynamics calculation. In order to complete the definition of motion dynamics for an object, however, you also need to describe the **forces** that act upon the object. What causes the object to move in the first place?

Perhaps the most basic force that motion dynamics takes into account is the force of **gravity**. Except in very rare circumstances, gravity affects all objects. Even if no other forces cause an object to move, gravity will keep it on the ground. Technically speaking, gravity is the attraction between two objects, caused by the masses of the objects and the distance between them. A ball falls to Earth because the mass of Earth is so much greater than the mass of the ball that the ball is attracted to Earth. This is why a ball in Antarctica will fall "upward" toward the Earth.

From the point of view of most animation, however, you can think of gravity in simpler terms—as a force pulling all objects downward. Consequently, most motion-dynamics systems define gravity as a simple force that pulls all objects downward at a specific rate in the negative Y direction. This force usually exists by default (just as it does on Earth) and affects all objects equally and automatically. Thus, even if you define no other forces and define no other animation, a ball that starts out suspended in the air at frame 1 of a simulation immediately begins falling in frame 2 because of gravity.

The fact that gravity causes objects to fall at a specific rate affects the dimensions you use when modeling objects in a motion-dynamics system. Because the speed at which an object falls is measured in terms of the distance the object travels per second, the measurements of distance you use in the modeling process must be consistent with the measurements of distance used by the motion-dynamics calculations for gravity. If the software you use measures gravity in feet per second, then the units of measurement for your models must also be feet. If gravity is measured in meters per second, then your models must be modeled in metric units. Normally, a motion-dynamics software package sets up default units of measurement, so that the units are consistent in both the modeling and simulation sections of the software. However, if you change the units of measurement, the gravitational force of the motion-dynamics system will not pull objects downward at a physically accurate rate.

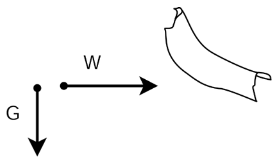


Figure 4-16. Motion-dynamics systems require you to define the forces, such as gravity and wind, that act on objects.

In addition to gravity, another force that many systems allow you to define is **wind**, which can be very useful, for example, in animating a curtain blowing in a breeze, or the branches of a tree bending slightly in the wind. These are subtle kinds of movement, difficult to model convincingly with keyframing techniques. A wind force is usually defined as a vector having a location, a direction, and a strength. In Figure 4-16, the arrow pointing to the right represents the wind force. The arrow pointing downward is the default gravity force. The curtain model has been defined to have a certain mass. When the wind force hits the curtain, it tends to push the curtain to the right. At the same time, however, the gravity force tends to pull the curtain straight down. Both of these tendencies are affected by the mass of the curtain itself. The result of these forces pushing and pulling against each other is that the curtain rises and falls and flaps "in the breeze."

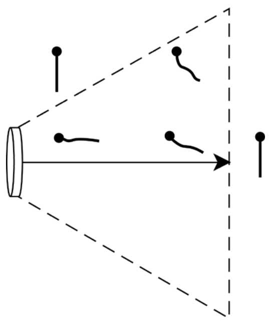


Figure 4-17. Unlike wind, which is omnipresent, a fan force has a limited range.

A wind force is defined in most systems to affect an entire scene equally from a given direction. For example, several curtains in a scene, each located some distance from the others, are affected to the same degree by the wind.

Some motion-dynamics systems offer a variation on the wind concept, called a **fan** force. Unlike wind, it usually is possible to limit both the distance over which a fan force will carry and the radial area that it will affect. In Figure 4-17, the fan force is represented by the cylindrical form on the far left. The arrow emanating from the center of that form and traveling to the right represents the distance over which the force of the fan will carry. At the base of the arrow—that is, at the cylin-

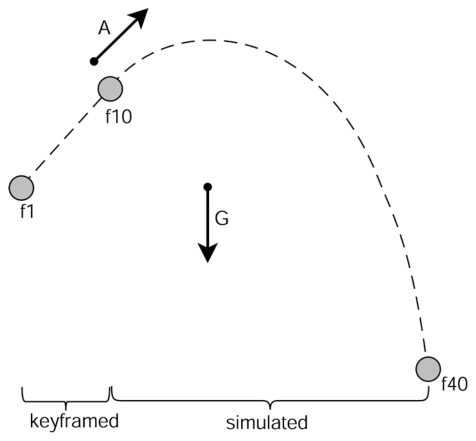


Figure 4-18. Some systems allow you to combine standard keyframed motion with motion-dynamics simulation. Here, the keyframed acceleration of an object is calculated into the motion dynamics.

drical fan icon itself—the force of the fan is strongest. This is why the string close to the base of the arrow is being blown quite a bit. As the force of the fan travels down the arrow, the strength of that force decreases, which is why the next string is not being blown as much as the first. Beyond the tip of the arrow, the force of the fan dies out completely and the string does not move at all.

The force of the fan not only fades as it travels in the direction of the arrow; it also fades as it moves radially away from the center of the fan. The dotted lines emanating from the fan icon demarcate the radial range of effectiveness. The string at the top left, which is completely outside the radial range of the fan, is not moving at all. The string directly to the right lies just inside the radial limit and therefore is being affected slightly. Notice, however, that this string is not being affected as much as the string directly below it, which, even though it is at the same horizontal distance from the fan, is closer to the radial center of the force field.

A final force that can be very important in setting up a motion-dynamics simulation is the acceleration an object has as the result of some keyframed action: **keyframed acceleration**. Acceleration is a measure of how fast the speed of an object changes, and the most basic way of giving an object some sort of speed or movement in a three-dimensional animation system is keyframing (described in chapter 3).

Some software packages allow you to combine keyframed animation movements with motion-dynamic simulation. Thus, you might keyframe an animation of a ball translating from one position to another over ten frames (Figure 4-18). This operation involves the standard keyframing techniques of setting a keyframe for the ball as it appears in

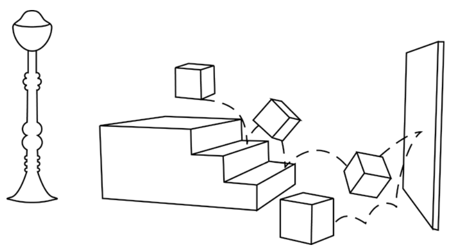


Figure 4-19. Only those objects that the falling box might hit need to be defined as obstacles for that box. This reduces the calculations necessary for collision detection.

frame 1, then setting another keyframe for the ball as it appears in frame 10. Having defined the keyframed animation from frame 1 to frame 10, you then direct the simulation software to take over the calculations from frame 11 through frame 40. When the simulation takes over, two forces are at work: the default gravity force, and the acceleration that the ball has as a result of being translated from the position at frame 1 to the position at frame 10. The combination of this keyframed acceleration and gravity causes the ball to continue rising for a while and then to fall.

One of the areas in which motion dynamics can be most useful is in simulating **collisions**. The reason the animation of a box falling down a flight of steps (see Figure 4-11) is so complex, for example, is that the box collides with the steps and bounces off them when it does. It is the precise calculation of these collisions, called **collision detection**, that makes a simulation of this animation so effective.

The calculation of collisions, however, can be extremely time-consuming for a computer system, and most motion-dynamics systems therefore allow you to elect whether or not to calculate collisions for a given object. If you know that an object, such as an isolated curtain (see Figure 4-16) won't hit anything in a particular animation sequence, you can turn off collision detection for that object.

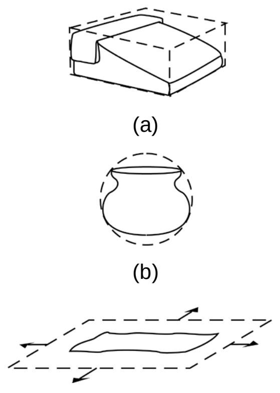


Figure 4-20. Bounding boxes, bounding spheres, and bounding planes—all represented by broken lines in this illustration—can be used to simplify collision-detection calculations.

If you want to calculate collisions for your model, however, most systems offer several options, all intended to minimize the amount of calculation necessary for a successful simulation. The first option is to define which other objects in the scene might be **obstacles**—that is, which objects the animated object might collide with. When the software does the collision-detection calculations, it will consider only these defined obstacles.

For example, suppose you add a lamppost to the scene in which the box falls down the steps (Figure 4-19). Given the position of the lamppost behind the steps, the box will never collide with it, so you can eliminate it from the list of obstacles that the motion-dynamics software needs to consider in collision-detection calculations. Only the steps, floor, and wall need be considered obstacles.

Another area in which most systems offer collision-detection options is in the shape of the objects involved. The collision calculations for a complex shape—a telephone, for example—are more complicated than the calculations for a simple shape, such as a box. Consequently, many systems allow you to use a simpler shape as a stand-in for the purposes of collision calculations. Assuming that a falling telephone is shaped like a simple box, for example, greatly speeds up the collision-detection calculations, although you still see the actual telephone falling. In many animations, the accuracy of the collisions calculations does not need to be exact in order to be visually convincing.

Among the number of simplifying shapes that motion-dynamics systems permit is the **bounding box**, or the smallest rectangular box into which an object will fit (Figure 4-20a). If you use the bounding box of the telephone for collision-detection calculations, the resulting motion is similar to what it would have been if you had used the actual shape of the telephone, since the shape of the bounding box and the shape of the telephone are similar. Likewise, some objects can be conveniently and fairly accurately approximated with a **bounding sphere**, or the smallest sphere into which they will fit (Figure 4-20b).

Bounding boxes and bounding spheres may be used to simplify either the colliding object itself or the obstacles with which the object will come into contact. Sometimes it is possible to use a **bounding plane** as a simplification technique for obstacles. A flat surface that extends infinitely in all directions, a bounding plane could successfully approximate a bumpy surface, for example (Figure 4-20c).

So far the examples in this section have involved rather simple models, but motion dynamics can be applied to more complex models and scenes as well. Models organized into a hierarchy can be part of a motion-dynamics simulation, as can models defined with inverse kinematics. In fact, some of the most impressive simulations involve precisely this combination of inverse kinematics and motion dynamics. You might model the blowing curtain, for example (see Figure 4-16), as an envelope over an inverse kinematic chain (as described in *Inverse Kinematics*, in chapter 3). When a motion-dynamics force acts on this model, the force affects each of the various links of each inverse kinematic chain (Figure 4-21a). If the bottom portion of a chain is blown by a wind force, for example, then the movement of the bottom portion of the chain causes the upper portions of the chain to move, and so on (Figure 4-21b). Since the envelope surface of the curtain is in turn controlled by the movements of the chains, the cur-

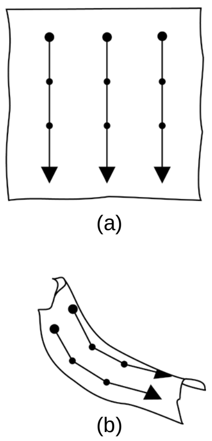


Figure 4-21. An inverse kinematic model can be effectively animated within a motion dynamics system.

tain animates, blowing in the wind. This combination of motion dynamics and inverse kinematics can be used to create very realistic and complex simulations.

Finally, it is important to understand one aspect of the underlying mathematics of motion dynamics. Whereas in the real world time is continuous, so that there is no break between one instant and another, in motion dynamics time is treated discretely. That is, time is broken up into a fixed number of measurable increments—for example, thirtieths of a second. These increments are called **time steps**. The smaller the time steps, the more closely they approximate the continuous nature of time, and therefore the more accurately they simulate an animated scene. However, smaller time steps also mean more calculations. Consequently, some systems allow you to define the size of the time steps, giving you the option of trading off accuracy of simulation for speed of calculation if necessary.

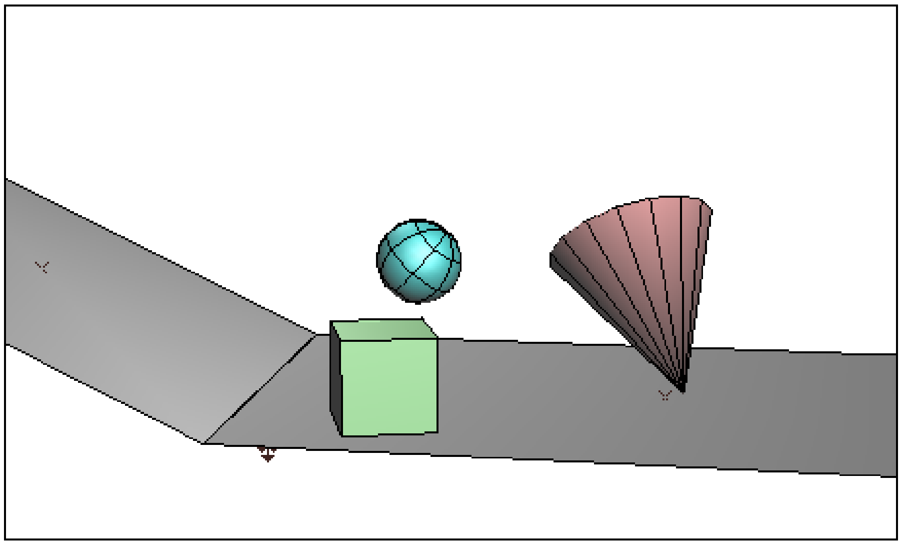
Because of the mathematics of motion-dynamics calculations, however, two simulations of a given scene that use different time steps will not produce the same motions. The path of an object can differ substantially when calculated under two different time steps. For this reason you may not find it useful to define a larger time step to sketch out a preliminary version of an animation, with the intention of switching to a smaller time step for the final animation.

All these motion-dynamics techniques are very powerful and can yield stunningly realistic simulations of an event. Sometimes, however, the most effective thing you can do is to combine the more standard, keyframe-based animation techniques with these sophisticated motion-dynamics calculations. For example, the human figure in color plate 2 was animated using standard keyframing and hierarchical animation. The bowling balls, on the other hand, were animated using motion dynamics to achieve a high degree of realism as the balls flew through the air and bounced on the ground about the feet of the character.

Exercise 1

Rigid Bodies, Collisions, Playback

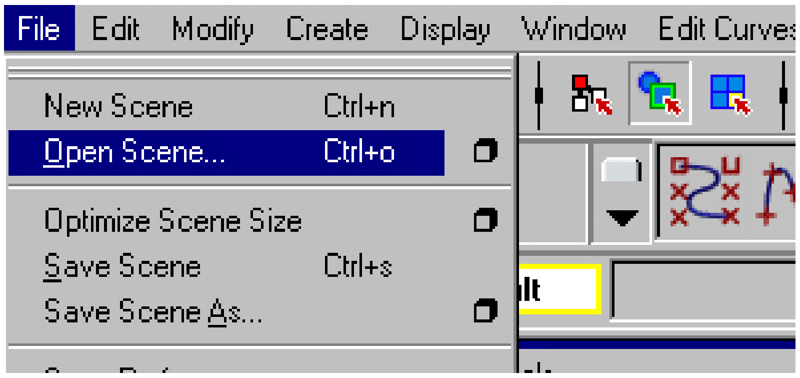
In this exercise, we will create a simulation in which several objects drop under the force of gravity. As they drop, they will collide with each other and other objects. Next, we'll adjust the objects' physical properties. Finally, we will create permanent animation curves for the simulation.



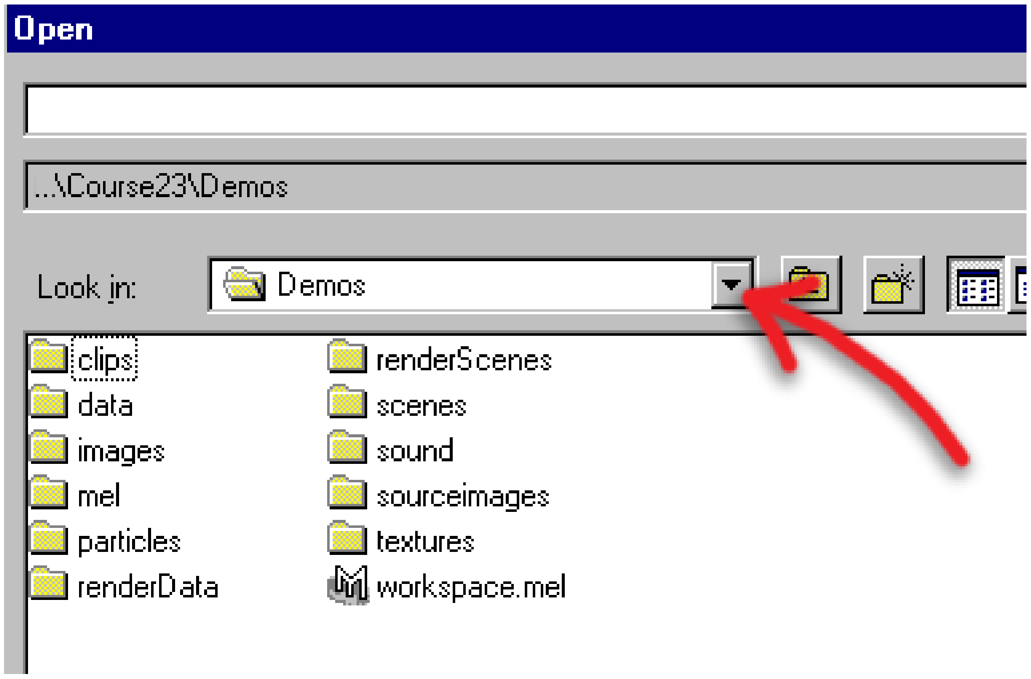
1. Set up your scene

Retrieve the starter scene file

To get started, retrieve the scene file which has been prepared for this exercise. This file contains the models necessary for the exercise, but no animation or simulation. To retrieve it, use
>File >Open Scene>



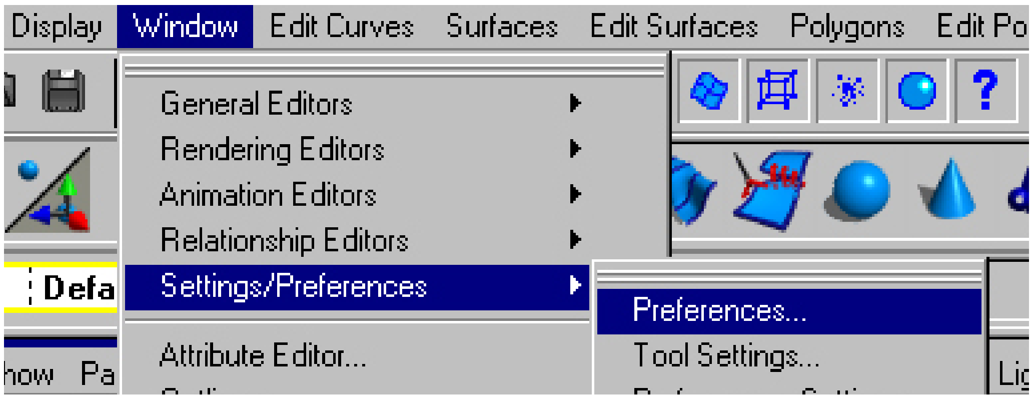
Inside the window that opens up, we must first locate the folder where the scene file is stored. Browse to find the folder *Course23/Demos.*



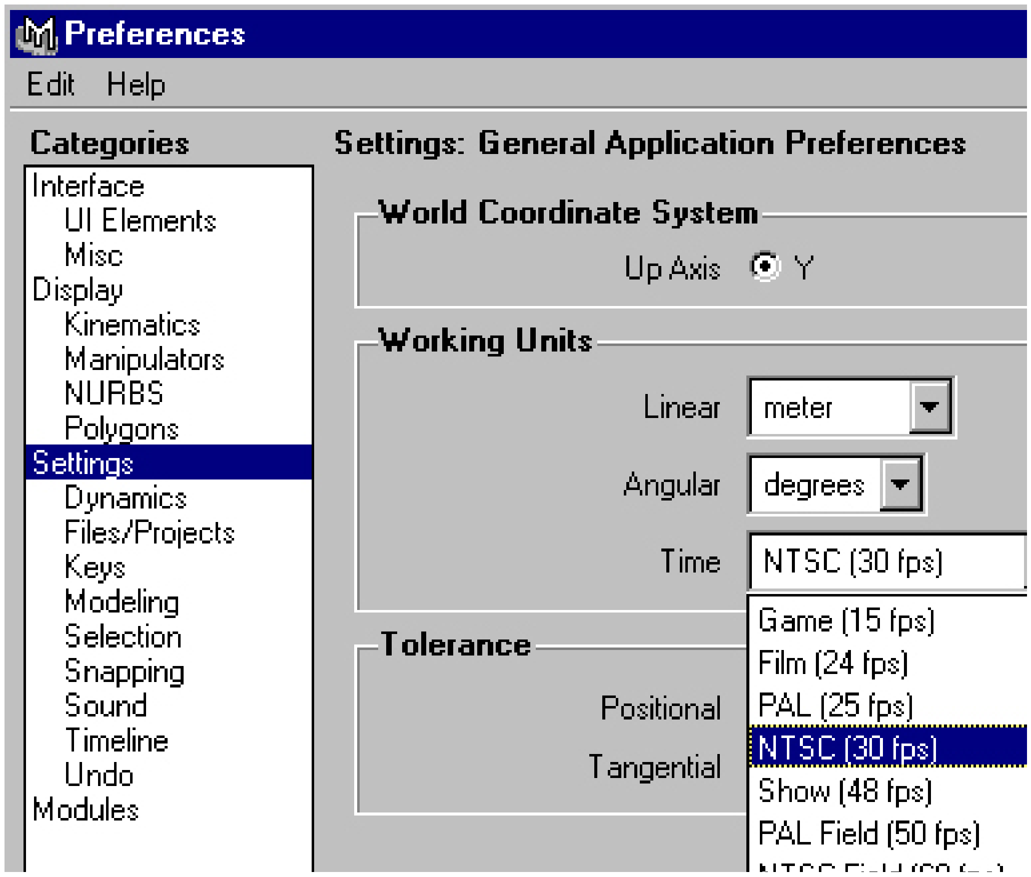
Double-click on the *scenes* folder and, from that folder, open the scene file called *Exercise1-Start.*

Define some important parameters

It is important to set certain parameters before you begin working. First, use **>Window >Settings/Preferences>Preferences.**



Inside the window that opens up, click Settings on the left side. On the right side, set Time = NTSC (30 fps). This is the standard video playback rate.



Settings: General Application Preferences

World Coordinate System

Up Axis [✓ Y]

Working Units

Linear: meter

Angular: degrees

Time: NTSC (30 fps)

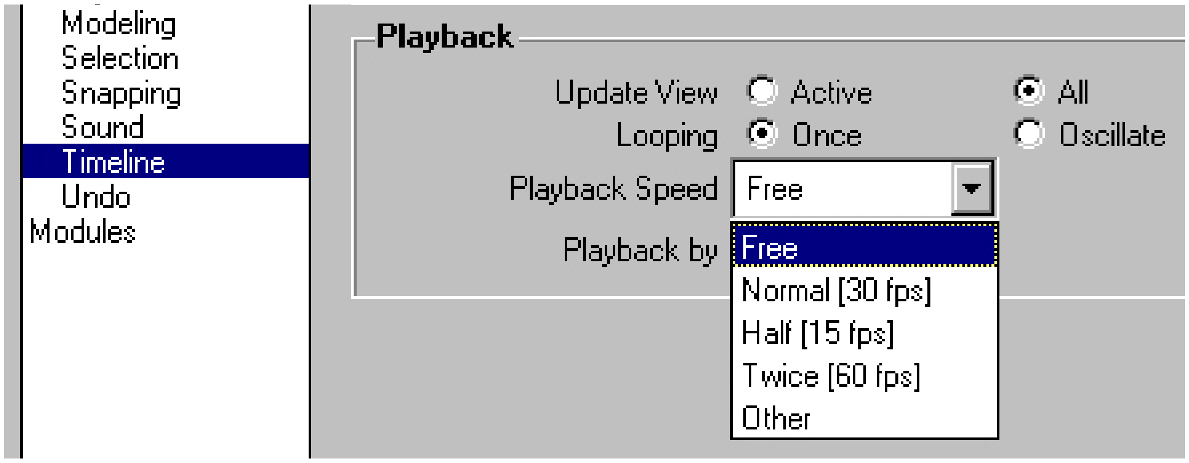
Tolerance

Positional

Tangential

Now click Timeline on the left side of the window. On the right side, next to Playback Speed, click Free. This will force Maya to play every frame. This is critical in order to get correct simulation calculations.

In the same window, next to Update View, click All, to cause Maya to play the animations in all open windows, rather than only in the selected window.



Maya windows

Familiarize yourself with navigating in the Maya windows.

While holding down the Alt key on your keyboard, press the middle mouse button and drag in any window. This tracks through that window.

Still holding down the Alt key on your keyboard, press the left mouse button and drag in the Perspective window. This tumbles (rotates) the camera in that window. Tumbling the camera is only possible in the Perspective window.

While holding down the Alt key on your keyboard, press and hold first the left mouse button then the middle mouse button, and drag left-right in any window. This dollies in and out in that window.

You can zoom in on any section of a window by holding down the Ctrl and Alt keys simultaneously and then using the left mouse button to drag a window around the region you want to see.

When you opened the *Exercise1-Start* scene file, it opened up four windows. The window on the lower right, with the black background and small icons, is called the Hypergraph window. It provides a diagram of the elements and nodes of your scene.

2. The Ball

We will begin by making the small ball drop and bounce off the ramp and floor.

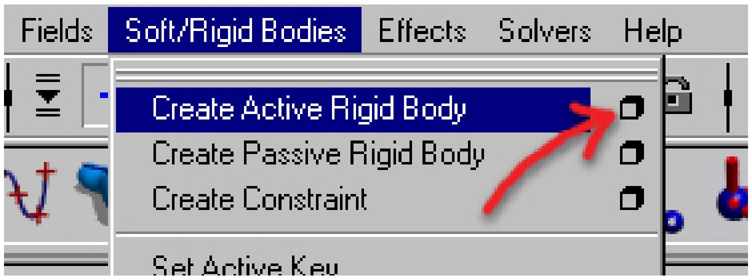
Go to the *Dynamics* module by hitting the F4 key.

Make the small ball an active rigid body

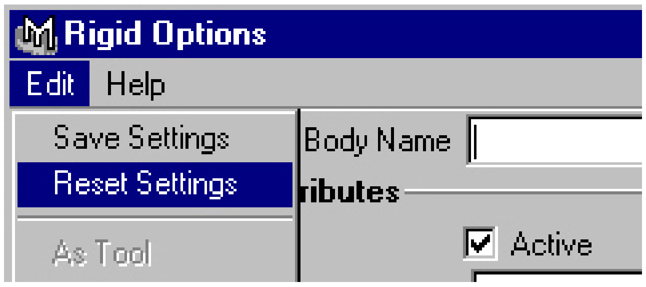
In Maya, a "body" is an object which is part of the motion dynamics calculations. An "active" body is one which can be moved around by motion dynamics. A "rigid" body is one whose shape does not change.

Select the ball by dragging your cursor over it, or by clicking on it. You can also select it by clicking on the icon of that name in the Hypergraph window.

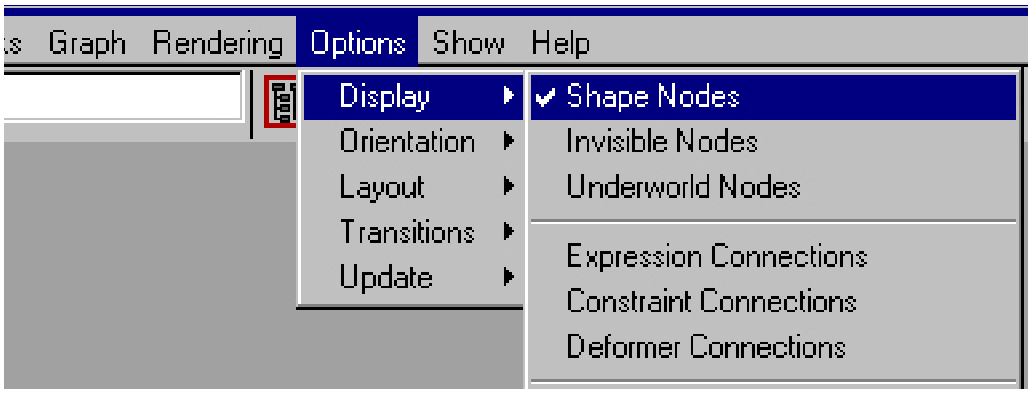
To define this ball as an active rigid object, use
>Soft/Rigid Bodies >Create Active Rigid Body []



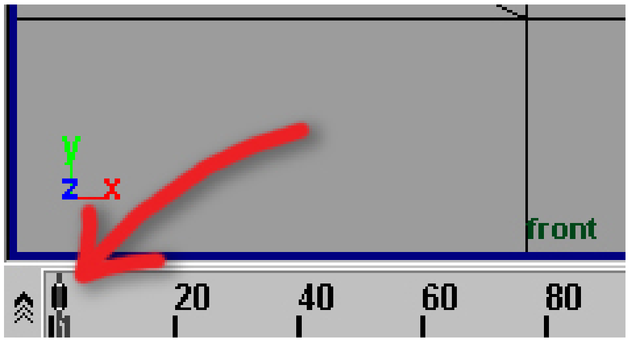
We will begin by using all the default values for the ball's physical properties. Make sure all values are set to the defaults with
>Edit >Reset Settings
then, hit **>Create**, to make the ball an active rigid body.



In the Hypergraph window, click on **>Options>Display>Shape Nodes** . This will display some additional nodes, or entities, below the icon of each object. Notice that the ball now has a rigid body node under it.

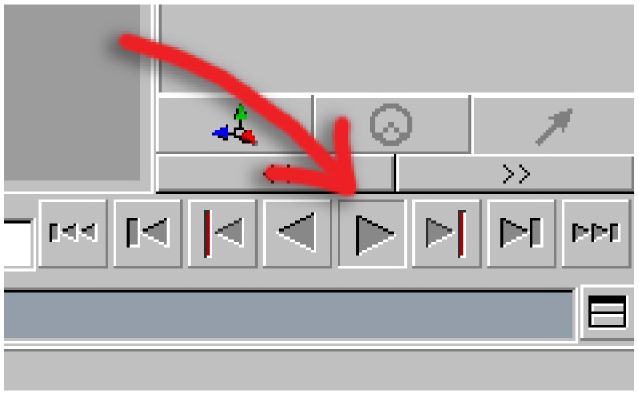


In the animation timeline at the bottom of the screen, drag the frame counter to frame 1.



This is important. You must always start your simulation playback at frame 1, in order to get accurate calculations of the simulation.

Hit the play button of the animation playback menu.

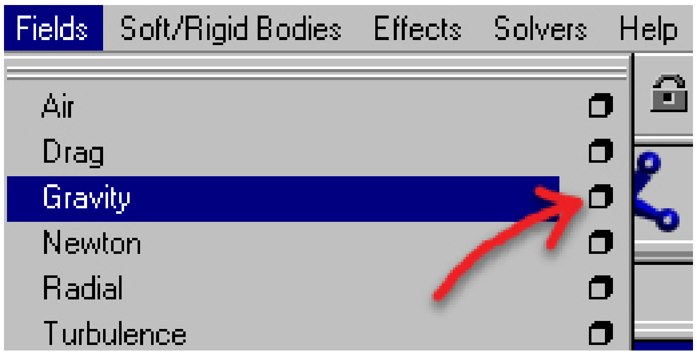


Nothing happens, because there are no forces acting on the ball. Stop the playback by hitting the little red square button in the animation playback menu.

Add Gravity

We will create a default gravity force, to cause the ball to drop.

Make sure the ball is still selected. (If it is not, drag the cursor over it to select it.) Now use
>Fields >Gravity[]



>Edit >Reset Settings, to use the default settings.
>Create

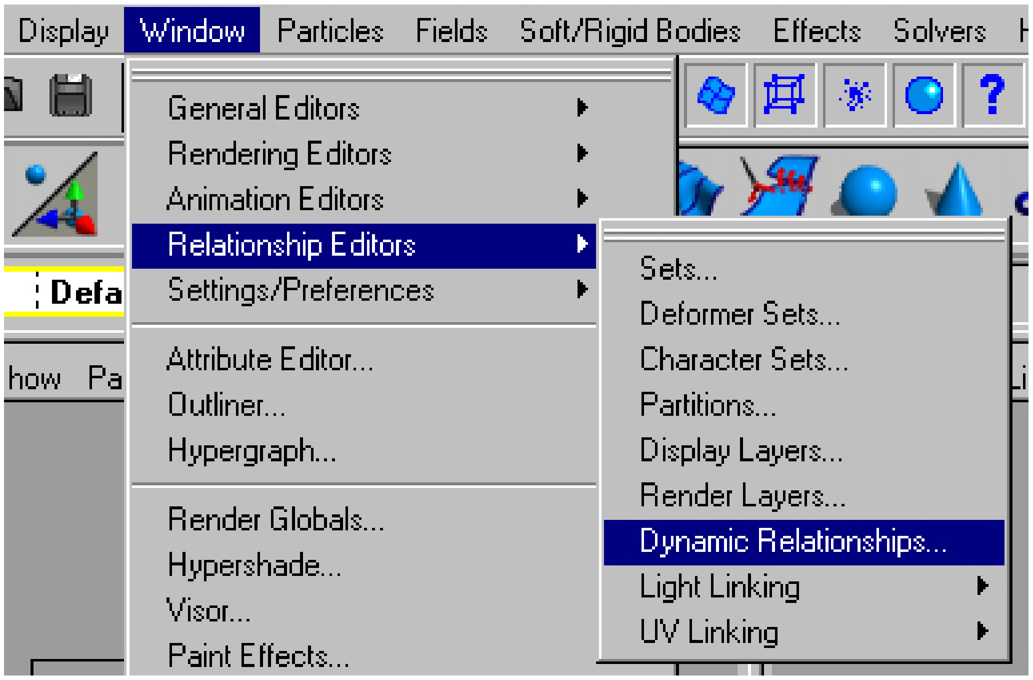
This creates a gravity force. Since the ball was selected when you created the gravity force, the gravity force is automatically attached to the ball – that is, the ball will be affected by the gravity force.

Drag the frame counter to frame 1. Hit the play button of the animation playback menu. Now the ball drops. But it goes through the ramp, because we have not yet defined any obstacles.

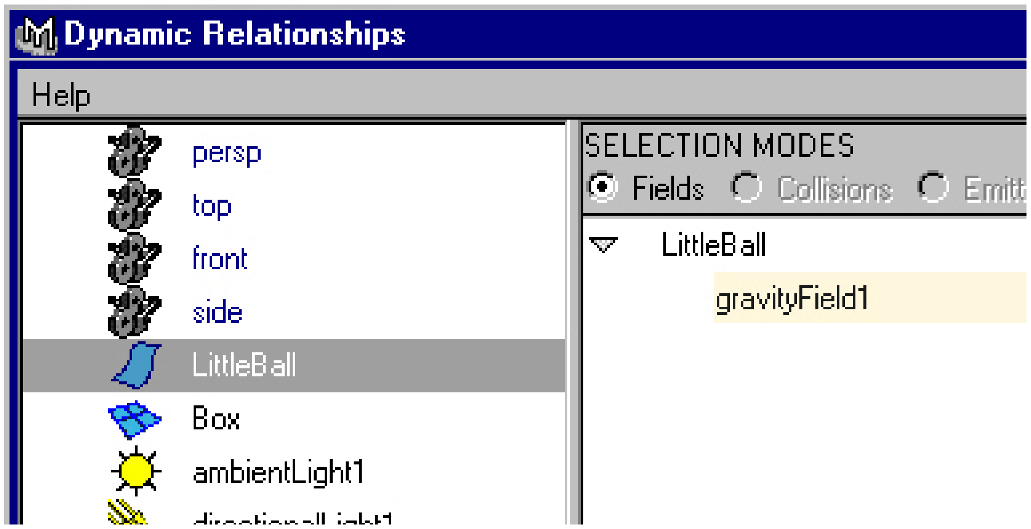
Stop the playback on the animation playback.

The Dynamics Relationship window

There is an important window in Maya which allows you to connect and disconnect forces to objects. To confirm that gravity has indeed been attached to the ball, use
>Window >Relationship Editors >Dynamic Relationships



In the left window, click on the name of the ball. In the right window, make sure Selection Mode is set to Fields.



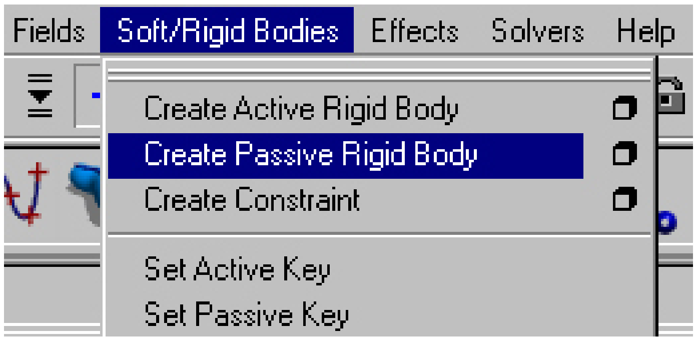
In this window, you should see *gravityField1* listed under the ball. This gravity field should be highlighted. If it is highlighted, that means it is connected to the ball. If *gravityField1* is not highlighted, click on it, to highlight it.

Define Obstacles

We will define both the ramp and the floor as obstacles. In Maya, an object which is part of the motion dynamics calculations, but which cannot be moved by motion dynamics, is called a "passive" body. An obstacle object is a "passive rigid body".

Select both the ramp and floor by dragging your cursor over both of them. You could also shift-click on each of them.

With the ramp and floor selected, use
>Soft/Rigid Bodies >Create Passive Rigid Body []



>Edit >Reset Settings
>Create

This makes both the ramp and floor obstacles.

Drag the frame counter to frame 1. Play the animation now. The ball should drop, and bounce off the ramp and the floor.

Adjust Bounciness

Any of the physical properties of the motion dynamics bodies can be adjusted. This is true of both active bodies and passive bodies. The easiest way to do this is with a window Maya calls the Attribute Editor.

We will adjust the bounciness of the ball. Select the ball. With the ball selected, use
>Window >Attribute Editor. (You can also show the Attribute Editor window at any time by simply hitting Ctrl-a on the keyboard.)

In the Attribute Editor, select the *rigidBody* tab. Inside that sub-menu, increase the *Bounciness* parameter to 0.8.) Drag the frame counter to frame 1 and play the animation. The ball should bounce much more than previously.

Open the Attribute Editor for the ball again. (Make sure the ball is selected.) Now change *Bounciness* to 0.0. Reset to frame 1 and play the animation. The ball should not bounce at all off the ramp.

Save your scene by using
>File >Save Scene As

All our exercise files will be stored in a folder called *Course23/Practice.* This should be already set as the default project folder. If it is not, browse to find that folder. All scene files are stored in a sub-folder called *scenes*.

Give your scene a name that will be unique and easy to remember. For example, your own name plus the exercise number.

Type in the name you want to use, then hit Save.

3. The Box

Now we will bring the box into the motion dynamics calculations. We will make it slide down the ramp.

Make the box an active rigid body

To make the box an active rigid body, first select the box. Then use
>Soft/Rigid Bodies >Create Active Rigid Body []
>Edit >Reset Settings
>Create

Attach gravity to it

Play the animation. The box doesn't move, because there are no forces acting on it.

Even though we have already defined a gravity force, it is not attached to the box. To attach it, use
>Window >Relationship Editors >Dynamic Relationships
In the left window, click on the name of the box. In the right window, make sure *Selection Mode* is set to *Fields.* In the right window, click on the gravity field underneath the box.This will attach gravity to the box.

Reset to frame 1 and play the animation. Now the box is pulled down by gravity, and starts to slide down the ramp.

Notice that we did not have to re-define the ramp and floor as passive rigid bodies (obstacles). Once you have defined an object as a passive rigid body, it is an obstacle for any and all active rigid bodies in the motion dynamics simulation.

Adjust Friction

Static friction is the friction which prevents an non-moving (static) object from starting to move. If static friction is very high, the box will not slide at all. If it is very low, it will slide very easily.

Select the box. Open the Attribute Editor for the box with either Ctrl-a, or with **>Window >Attribute Editor**.

Click on the *rigidbody* tab at the top of the Attribute Editor window. Under *Friction*, change *Static Friction* to 1.0. Play the animation. The box should not slide down the ramp at all. (If the box does not slide, it is probably because it is intersecting the ramp. Select the box, go into *Move* mode by hitting the w key on your keyboard, and move the box very slighty above the ramp.)

Back in the box's Attribute Editor, change the static friction to 0.0. Play the animation. The box should immediately slide down the ramp.

Save your scene by using
>File >Save Scene As
Remember that the project folder for your files is *Course23/Practice.* Browse to find that folder if necessary.

It is a good idea to save a new version of your scene file, rather than using the same name and over-writing your previous file.

Type in the name you want to use, then hit Save.

4. The Cone

We will now add the cone to the simulation calculations.

Make the cone an active rigid body

Select the cone and use
>Soft/Rigid Bodies >Create Active Rigid Body

Attach gravity to it

Just as you did with the box, attach gravity to the cone by using
>Window >Relationship Editors >Dynamic Relationships

Reset to frame 1 and play the animation. The cone drops and bounces off the floor.

Adjust the Center of Mass

Open the Attribute Editor for the cone, by selecting the cone and using Ctrl-a. Click its *rigidbody* tab to look at the attributes for the rigidbody node of the cone. Under *Center of Mass*, the three numbers represent the X, Y, Z location of the center of mass. Leave the X and Z values as is, but change the Y value to –2.0. This moves the cone's center of mass to its wide bottom.

Reset to frame 1 and play the animation.The cone should bounce differently now, because its center of mass is in a different place.

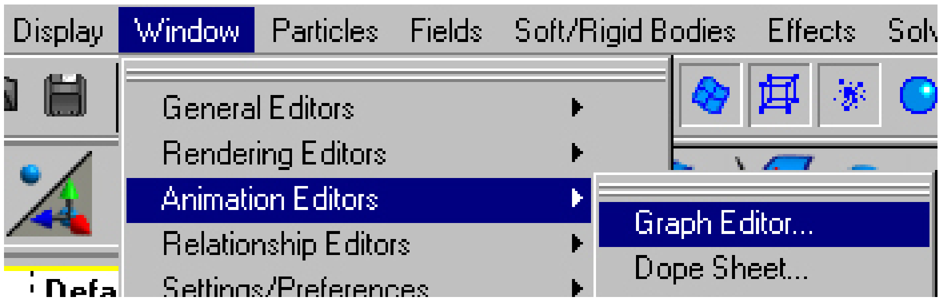
Try changing its center of mass again. Play the simulation.

Save your scene by using
>File >Save Scene As

5. Make the simulation permanent

By default, each time you play a motion dynamics simulation by hitting the Play button on the animation playback menu, Maya recalculates the movement for each object. That is, the animation information is not saved in any permanent form.

Select the ball. With the ball selected, use
>Window >Animation Editors >Graph Editor

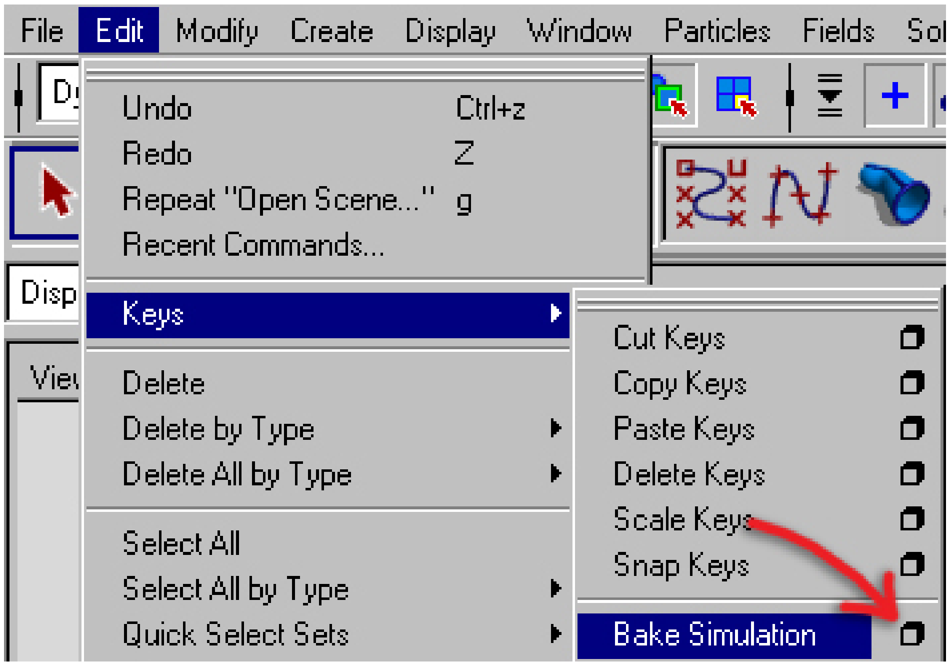


In standard animation (for example, keyframed animation), the motion of each object can be viewed as graphs, or curves. Notice that the *Graph Editor* which we just opened is showing that no curves at all have been created for the ball. Each time we play the animation, Maya recalculates the motion of the ball from scratch.

"Bake" the Ball's simulation

Maya's technique for storing permanent animation curves for an active rigid body is called "baking" (Funny name, but that's what they call it.)

Select the ball. Use
>Edit >Keys >Bake Simulation []



Make sure that *Time Range* is set to *Time Slider*. This will bake the ball's motion for all the frames in the animation. Click Bake.

Examine its function curves

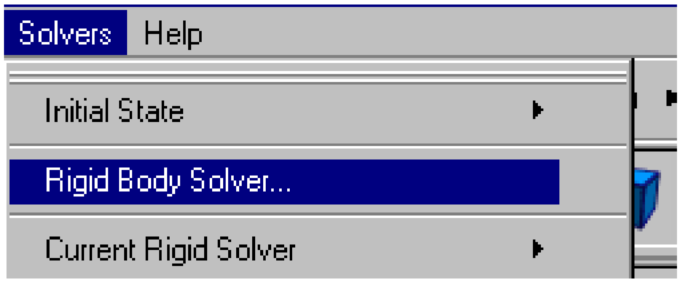
Notice that, after baking, the animation timeline at the bottom of your monitor is full of red lines. These represent the key frames that baking created at each and every frame of the ball's motion.

Use **>Window >Animation Editors >Graph Editor** to look again at the ball's animation curves. Now there are curves representing the ball's motion.

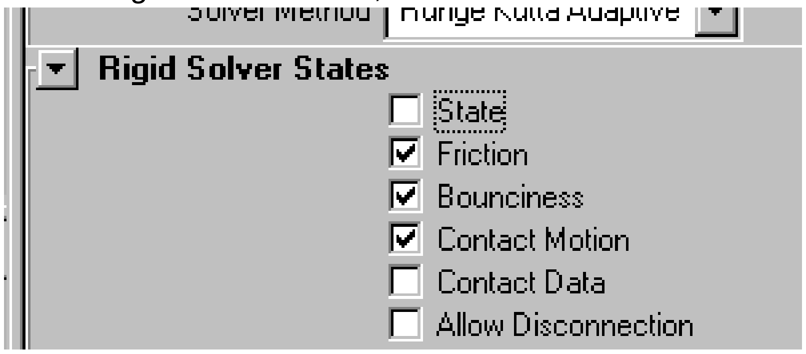
Playback without simulation

Since animation curves have now been calculated for the ball's motion, we no longer need to do a motion dynamics simulation to see that motion.

Open **> Solvers>Rigid Body Solvers**



Under *Rigid Solver States*, click *State* off.



☐ State
✓ Friction
✓ Bounciness
✓ Contact Motion
☐ Contact Data
☐ Allow Disconnection

This tells Maya NOT to use that solver. In other words, NOT to do the motion dynamics calculations.

Now play the animation. The only object that moves is the ball, because it is the only one for which we created animation curve information. All of the other objects don't move at all. Since they don't have curves, and since we told the rigid body solver not to do motion dynamics calculations, they have no movement at all.

Bake the other objects

Now we will bake the simulation for the other objects, just as we did for the ball.

First, turn the rigid body solver back on, to tell Maya to once again do the motion dymanics calculations. Open **>Solvers>Rigid Body Solvers**.

Under *Rigid Solver State*, Click *State* back to on.

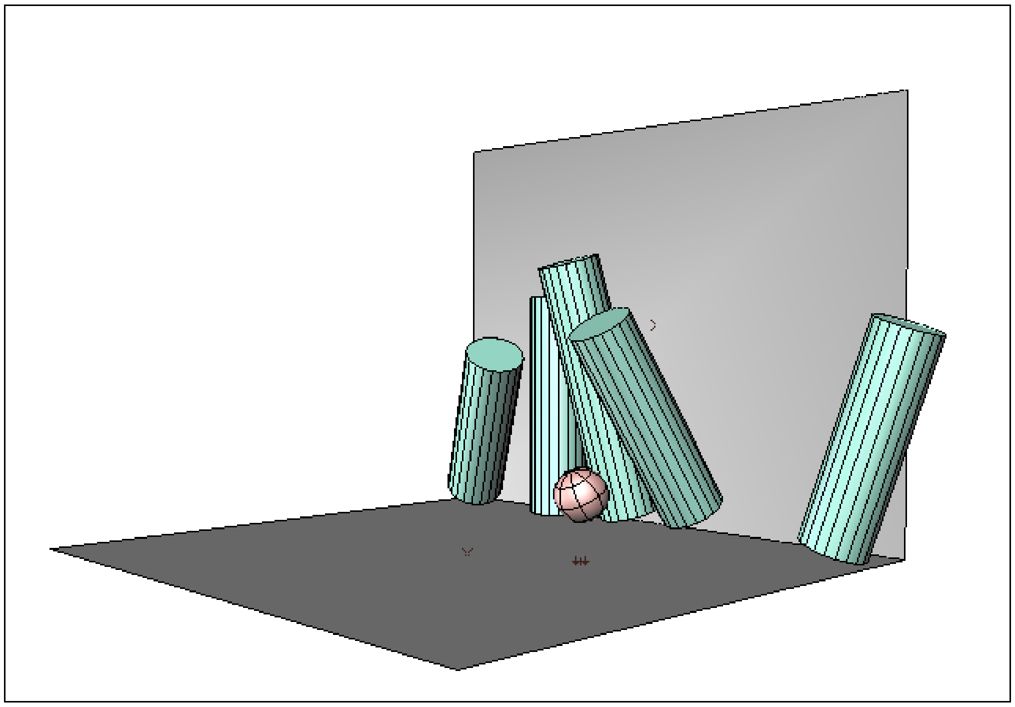
Select both the cone and the box. Use
>Edit>Keys>Bake Simulation, just as you did with the ball.

Save your file with **>File>Save Scene As**. Make sure to use a new filename, so you don't overwrite your previous file.

Exercise 2

Inherited Velocities

In this exercise, we combine keyframed animation with motion dynamics animation. We will keyframe the initial motion of a ball, and have the motion dynamics software calculate its remaining motion. The ball will crash into several objects, knocking them over. Lastly, we'll adjust several parameters to fine-tune the simulation.



1. Get the starter scene

Retrieve this exercise's starter scene with
>File >Open Scene

Remember that you must first locate the folder where the demo scene files are stored. Browse to find the folder *Course23/Demos.*

Inside that folder, go to the *scenes* sub-folder and open the scene file called *Exercise2-Start.*

2. The Cylinders

Make the cylinders active rigid bodies

Select all the cylinders by click-dragging your cursor over all of them. Alternatively, you can shift-click on each of them in succession. You could also select them by clicking on their icons in the Hypergraph window.

Use **>Soft/Rigid Bodies >Create Active Rigid Body**

Add Gravity

With all the cylinders still selected, use **>Fields >Gravity**

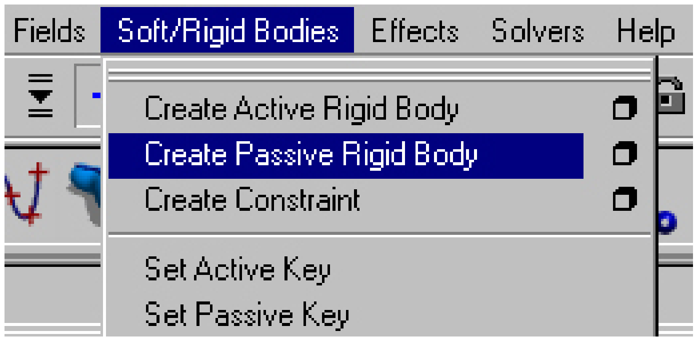
The gravity field you just created should be automatically attached to the cylinders, if they were selected at the time you created gravity.

Use **>Window >Relationship Editors >Dynamic Relationships** to make sure that each cylinder is attached to gravity.

3. The floor and wall

The floor and wall will be obstacles. Select them both, and use

>Soft/Rigid Bodies >Create Passive Rigid Body



Save your scene by using **>File >Save Scene As**.
Remember that all your exercise files get stored in the folder called *Course23/Practice*. This folder should be your default project. If it is not, browse to find that folder. Type in the filename you want to use, then hit Save.

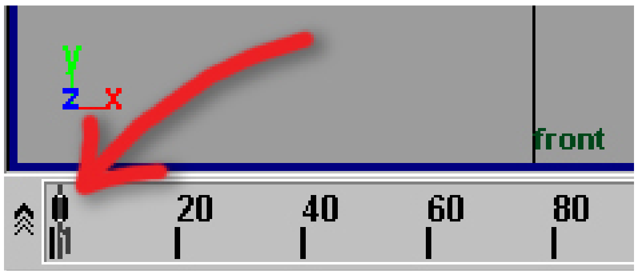
4. The Ball

In order to give the ball some initial motion, we will keyframe its motion in the first few frames. After these first few frames, the motion of the ball will be calculated automatically by the motion dynamics simulation.

Define the first keyframe for the ball

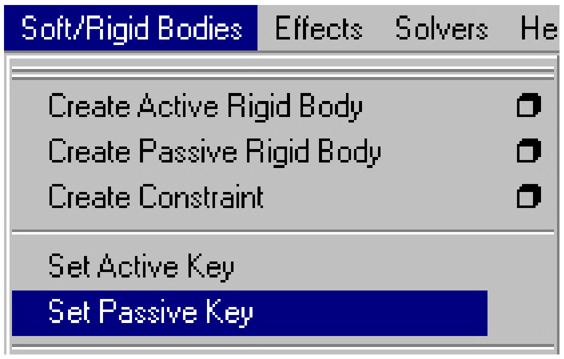
Select the ball.

Make sure time slider's frame counter at the bottom of the screen is dragged all the way to the left to frame 1.



With the ball selected and still in its original position (that is, to the left of the cylinders), use

>Soft/Rigid Bodies >Set Passive Key

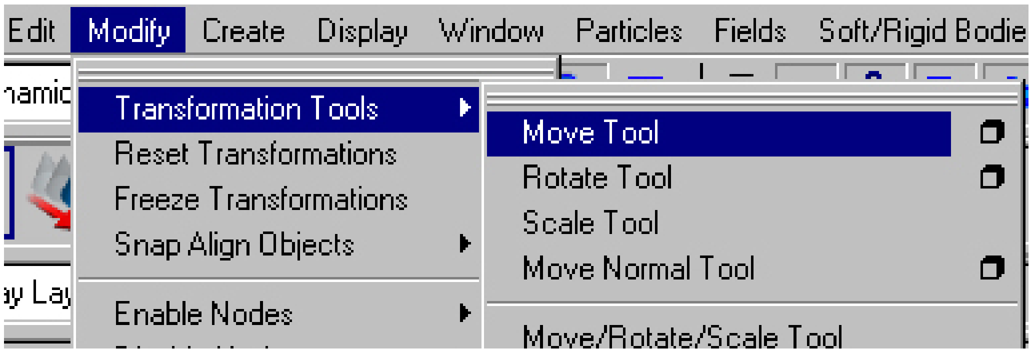


This keyframes the position of the ball at frame 1. It simultaneously makes the ball a rigid body.

Define the second keyframe for the ball

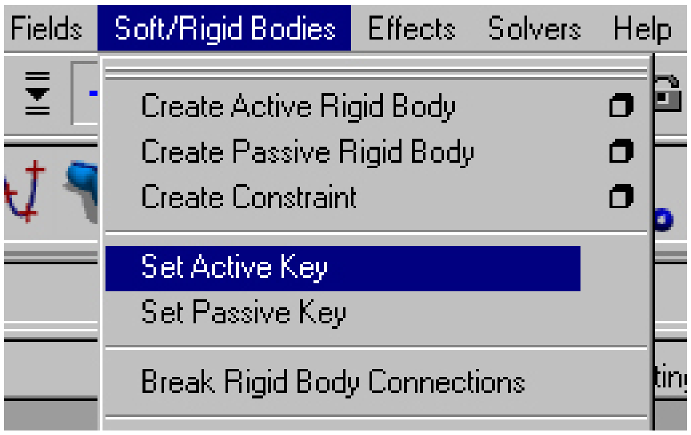
Now move the time slider's frame counter over to frame 5.

Now we will re-position the ball. Make sure the ball is still selected. Use **>Modify >Transformation Tools >Move Tool**
This puts you into Move (i.e., Translation) mode. An alternative way to go into Move mode is to simply hit the lower-case **w** key on your keyboard.



You will see an axis-like icon appear at the ball. Click on the red arrow and drag the ball toward the cylinders. Using the red arrow of the icon restricts your motion to movement only along the ball's X axis. The farther the ball travels in these first 5 frames, the faster it will be going at the end of frame 5. Try moving the ball over about 8 units towards the cylinders.

With the ball still selected and in its new position, and the frame counter at frame 5, hit
>Soft/Rigid Bodies >Set Active Key.



This tells Maya to keyframe the new position of the ball. It also tells Maya that from this frame on, the ball is "active" — that is, its motion will be calculated by motion dynamics.

Save your scene with **>File >Save Scene As**

5. Fine-tune the simulation

Play the simulation

Play the animation.

The ball shoots forward from frame 1 – 5 because of its keyframed animation.

From frame 6 on, the motion of the ball is calculated by the motion dynamics simulation. Since the ball was already moving very quickly at frame 5 (because of the keyframed motion), the ball's velocity is "inherited", and the motion dynamics calculations carry it forward – and it smashes into the cylinders.

Attach gravity to the ball

Notice that the ball keeps flying forever after hitting the cylinders. This is because there is no gravity on it to pull it down.

Attach gravity to the ball by using **>Windows >Relationship Editors >Dynamics Relationships**

Run the simulation again by hitting the animation play button.

Adjust the ball's mass

We will adjust the mass of the ball.

Select the ball. Open its Attribute Editor with **>Window >Attribute Editor** (You can also use Ctrl-a as a shortcut to get the Attribute Editor.)

Click on the tab for the *rigidbody* node.

For the first five frames of our animation, the ball is not an active body. Consequently, many its motion dynamics parameters are not adjustable in those first five frames. Advance the frame counter to a frame after frame 5. Now the ball's motion dynamics parameters can be changed.

Under the tab for the *rigidbody* node, change the ball's mass to a very small number. Try 0.01.

Run the simulation again by hitting the animation play button. Now the ball bounces off the cylinders. If the ball's mass is light enough, the cylinders hardly move at all.

Now use the Attribute Editor to change the ball's mass to a high number. Try 10.0.

Run the simulation again. When the ball is very massive, its motion is difficult to stop. Now, when it hits the cylinders, it smashes them very hard and the cylinders go flying all about.

Save your scene by using **>File >Save Scene As**

6. Bake the simulation

Using the same procedures we saw at the end of Exercise 1, "bake" the simulation. Remember, this entails selecting all of the active rigid bodies, and using
>Edit >Keys >Bake Simulation []

After baking the simulation, you can delete all the rigid bodies with

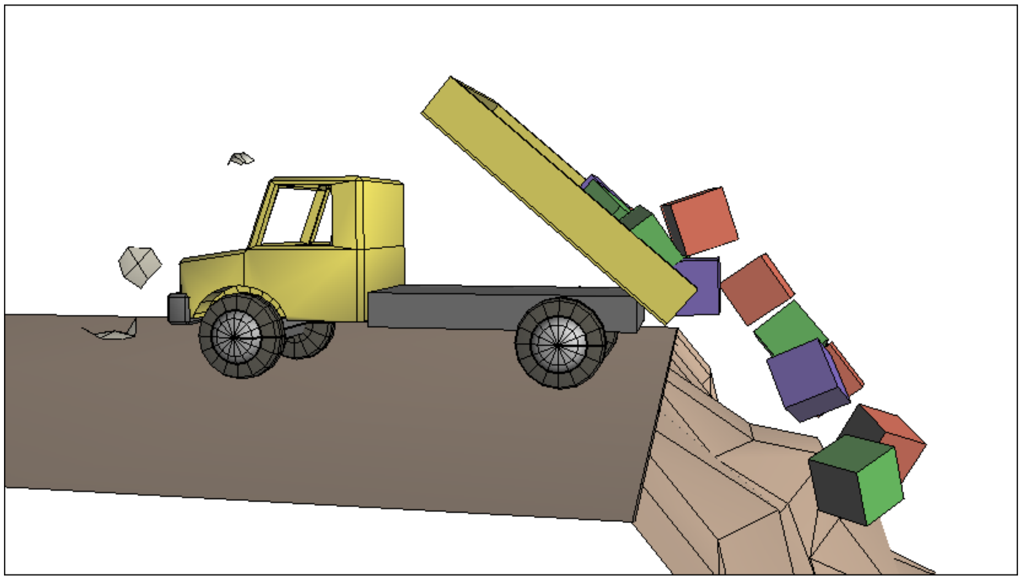
>Edit >Delete All by Type >Rigid Bodies. The objects which used to be rigid bodies will now be normal, keyframed objects.

Save your scene under a new name by using **>File >Save Scene As.** Make sure NOT to overwrite your previous version, since the new version you are saving has no motion dynamics in it anymore.

Exercise 3

A Rigid-body Animation Sequence

In this exercise, we will use the rigid body motion dynamics techniques we saw in Exercise 1 and Exercise 2 to create a simple animation scene. The scene consists of a dump truck at a (very simplified) dump. We will make the dump truck dump its load of boxes by keyframing the truck's bed, causing the boxes to tumble out. Finally, you will make some paper trash blow around with a gust of wind.



1. Get the starter scene

Retrieve this exercise's starter scene with
>File >Open Scene

Remember that you must first locate the folder where the demo scene files are stored. Browse to find the folder *Course23/Demos*.

Inside that folder, go to the *scenes* sub-folder and open the scene file called *Exercise3-Start*.

2. The Load of Boxes

Make the boxes active rigid bodies

Select all of the boxes in the back of the truck, by shift-clicking on each one. You can also select them in the *Hypergraph* window.

With all the boxes selected, use
>Soft/Rigid Bodies >Create Active Rigid Body
Use the default values for now. You will adjust them later.

Attach gravity to the boxes

With the boxes still selected, use
>Fields >Create Gravity

Play the simulation by hitting the animation playback Play button.
The boxes should all drop—right through the truck bed, in fact.

If the boxes do not drop, confirm that gravity is attached to them with
>Window >Relationship Editors >Dynamic Relationships

Save your scene by using
>File >Save Scene As
Remember that all your exercise files should be stored in the folder called *Course23/Practice*. If necessary, browse to find that folder. Type in a name for your scene, then hit Save.

3. The Truck Bed

We want two things to happen with the truck bed. First, we want it to be an obstacle, so that the boxes do not go through it. Secondly, we want to animate it, so that as it moves, the boxes will move with it.

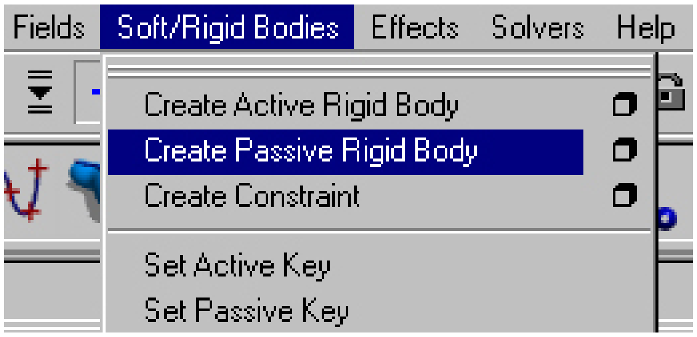
Both of these are simultaneously accomplished by setting "passive key" frames for the truck bed

Keyframe the truck bed's rotation

Select the truck bed. You can do this either by clicking on it in one of the modeling windows, or by selecting its icon in the *Hypergraph* window.

Make sure the time slider's frame counter (at the bottom of the screen) is on frame 1.

With the truck bed selected and still lying flat in its original position, use **>Soft/Rigid Bodies >Set Passive Key**



This defines the truck bed as a passive rigid body; and simultaneously keyframes its position at frame 1.

Now, advance the time slider's frame counter to frame 90.

We need to rotate the truck bed upwards. Make sure the truck bed is still selected.

An easy way to go into *Rotate* mode is to simply hit the *e* key on your keyboard. You should see an icon consisting of three circles appear. Click on the red circle and drag it until the truck bed rotates upwards. (The truck bed model has been built so it can only rotate in this one direction.)

Make sure you are still at frame 90. When you get the truck bed rotated the way you like it, hit
>Soft/Rigid Bodies >Set Passive Key
This sets a new keyframe at frame 90 for this passive rigid body.

Play the simulation by hitting the animation playback Play button.
As the truck bed rotates upward, the boxes slide off and fall into space.

Save your scene by using
>File >Save Scene As

4. The Terrain

We want the boxes to collide with the terrain. That is, we want the terrain surfaces to be obstacles, or, in Maya terminology, passive rigid bodies.

Since the truck is at the edge of the pit, we can predict that none of the boxes will collide with the flat surface that the truck is sitting on. To reduce the collision calculations therefore, we will only make the other two terrain surfaces obstacles.

Make the terrain pieces obstacles

Select the pit surface and the far-wall surface. Use
>Soft/Rigid Bodies >Create Passive Rigid Body

Play the simulation by hitting the animation playback Play button.
The boxes now slide off the truck bed and bounce off the terrain surfaces.

Save your scene by using
>File >Save Scene As

5. Fine-tune Pysical Properties

Adjust some of the physical properties by selecting an object and opening its Attribute Editor. To do this, you can use either **>Window >Attribute Editor**, or simply Ctrl-a.

Try making one of the boxes very massive. This will cause it to have more forces as it knocks the other boxes about.

Try making some of the boxes more or less bouncy.

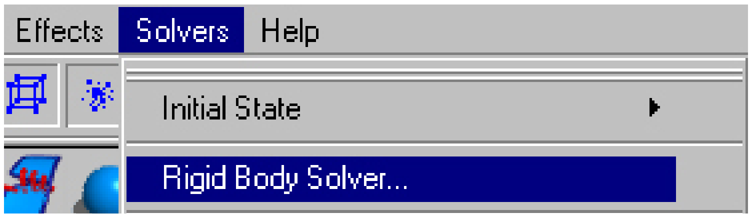
Save your scene by using
>File >Save Scene As

6. Change the precision of the calculations

It is possible for a simulation to calculate the movement of your objects inaccurately. For example, an object might pass through an obstacle. Or an object might pass part-way through an obstacle and then get stuck. This happens because the simulation calculations are not precise enough.

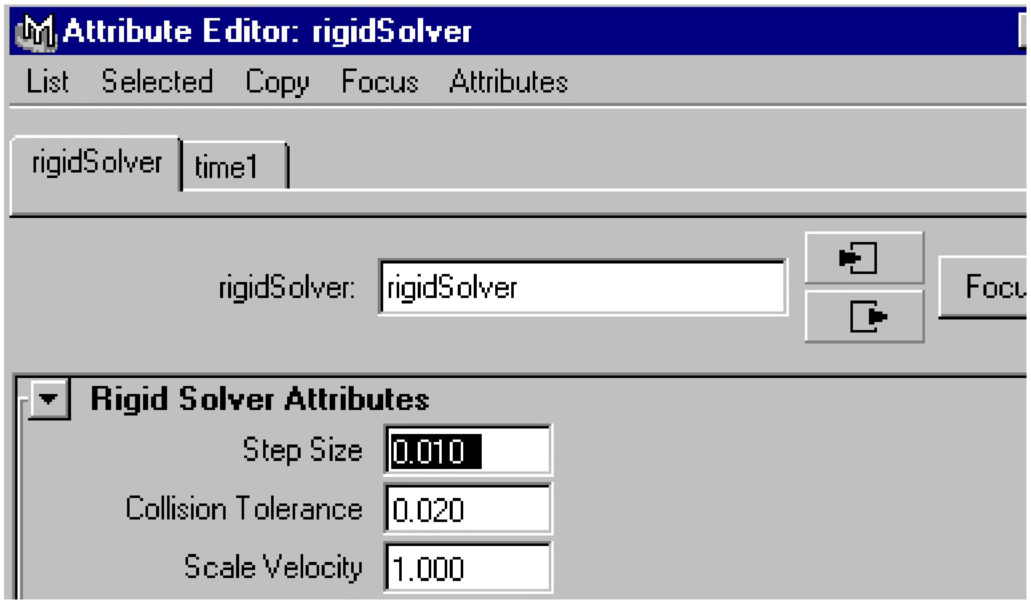
One parameter that determines precision is how frequently the calculations are done. In Maya, the default, is that the calculations are done about once every frame—that is, once every 30th of a second. If you make that time step smaller, the calculations will be more precise. They will also be slower, since they are being done more frequently.

Use **>Solvers>Rigid Body Solver**



Inside the window that opens, select the rigidSolver tab. Change the Step Size number to a smaller number. The default is 0.03. This is about 1/30th of a second. If necessary, change that number to a smaller number—for example, 0.01. Play the animation to see if the simulation improves.

Inside the Rigid Body Solver window, also look at Collision Tolerance. As this number becomes smaller, the collision calculations become more precise (and, of course, slower). If necessary, change that number to a smaller number.



rigidSolver: rigidSolver
Step Size 0.010
Collision Tolerance 0.020
Scale Velocity 1.000

Save your scene by using
>File >Save Scene As

7. Blow the Paper Trash Around

There are some paper trash objects lying on the ground near the front of the truck. You can enhance the trashiness of your dump by blowing the paper trash around with some wind.

You may want to temporarily hide the truck and boxes. To do so, select them and use **>Display>Hide>Hide Selected**.

Make the paper trash active rigid bodies

Select all of the paper trash objects. Use
>Soft/Rigid Bodies >Create Active Rigid Body

Create a turbulence force

A turbulence force is a force that pushes things around in a somewhat random way. This will simulate an irregular wind force.

>Fields >Turbulence []
>Edit >Reset Settings
>Create

If the paper trash objects were not selected when you created the turbulence field, you will need to attach the turbulence field to each of them. Use
>Window >Relationship Editors >Dynamic Relationships

Play the simulation by hitting the animation playback Play button.
The paper trash objects should blow around.

Attach gravity to the paper trash

Now we need to attach gravity to the paper trash objects, so they will not fly off into space. We have already created a gravity force for our boxes. We can use the same gravity force for the paper trash. Use
>Window >Relationship Editors >Dynamic Relationships
to attach gravity to each of the paper objects.

Add another terrain obstacle

The trash is not colliding with the flat part of the terrain, because that part was not originally defined as an obstacle. Select the flat part of the terrain that the trash is on. Then use
>Soft/Rigid Bodies >Create Passive Rigid Body
to make it an obstacle.

Play the animation. The trash should now bounce off that part of the terrain.

Save your scene by using
>File >Save Scene As

8. Fine-tune the Turbulence

Now we will adjust various parameters to fine-tune the way the tubulence blows the papers around. By default, the force of the turbulence field fades with distance. We will change this so that it does not do so.

Begin by selecting the turbulence field. This may be difficult to do in the modeling windows. Use the Hypergraph window (**>Window >Hypergraph**) to select the turbulence icon.

With turbulence selected, open the Attribute Editor (**>Window >Attribute Editor**, or Ctrl-a) for the turbulence field. Under Turbulence Field Attributes, change the Attenuation of the turbulence to 0.0. Play the simulation now. You may also need to adjust the Magnitude of turbulence.

Save your scene by using
>File >Save Scene As

9. Keyframe the Turbulence

Right now, the turbulence is "blowing" all the time. To simulate a gust of wind, we will turn the turbulence off, then on, then off again.

Drag the time slider to frame 1.

Make sure the turbulence is still selected. Open its Attribute Editor. Find the Magnitude parameter under Turbulence Field Attributes. Set the Magnitude to 0.0.

Now place your cursor over the word Magnitude. Holding down the right mouse button, drag to Set Key. This keyframes the Magnitude value at frame 1.

Drag the time slider to frame 100. In the Attribute Editor of Turbulence, change the magnitude to a large number—for example, 70.0. Set a keyframe by using the right mouse button over the word Magnitude, and then Set Key.

Now advance the time slider to frame 120. Change Magnitude back to

0.0. Set a key for it in the same way as before.

Play the animation. The trash should not blow around at first, then be blown very quickly, then die down again.

(To really fine-tune the animation on the Magnitude parameter, you could use the Graph Editor to change its interpolation type to "Stepped", which gives you an on/off quality.)

Save your scene by using
>File >Save Scene As

10. Bake the simulation

Use the same procedures we saw at the end of Exercise 1 to "bake" the simulation. Remember, this entails selecting all of the active rigid bodies—all the boxes, and all the paper trash objects—and using **>Edit >Keys >Bake Simulation []**

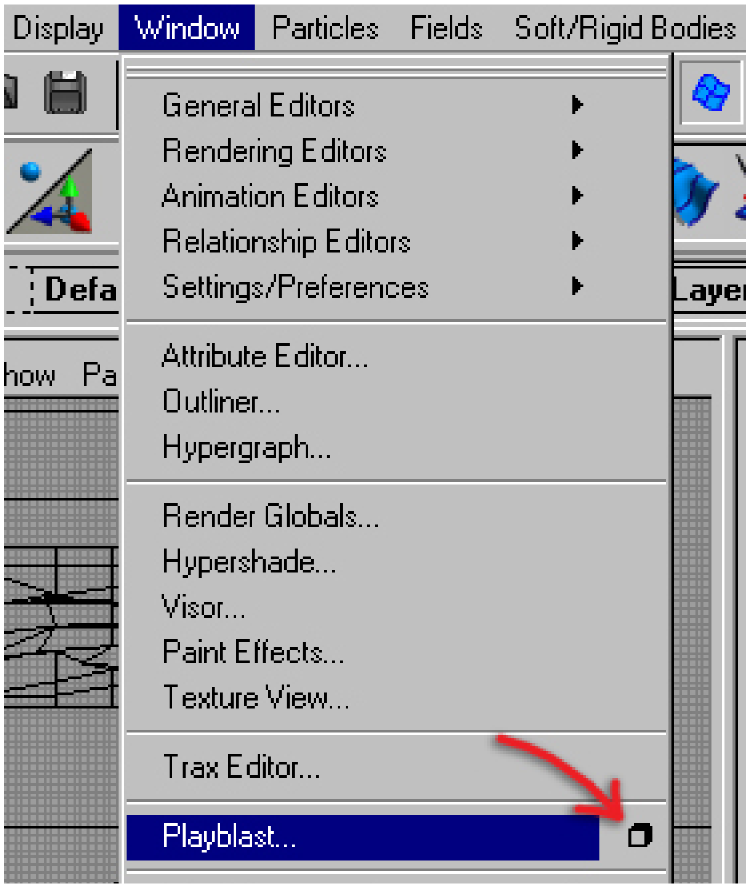
After baking the simulation, you can delete all the rigid bodies with **>Edit >Delete All by Type >Rigid Bodies**. The objects which used to be rigid bodies will now be normal, keyframed objects, and no motion dynamics calculations will be needed for playback.

Save your scene under a new name by using **>File >Save Scene As.** Make sure *not* to overwrite your previous version, since the new version you are saving has no motion dynamics in it anymore.

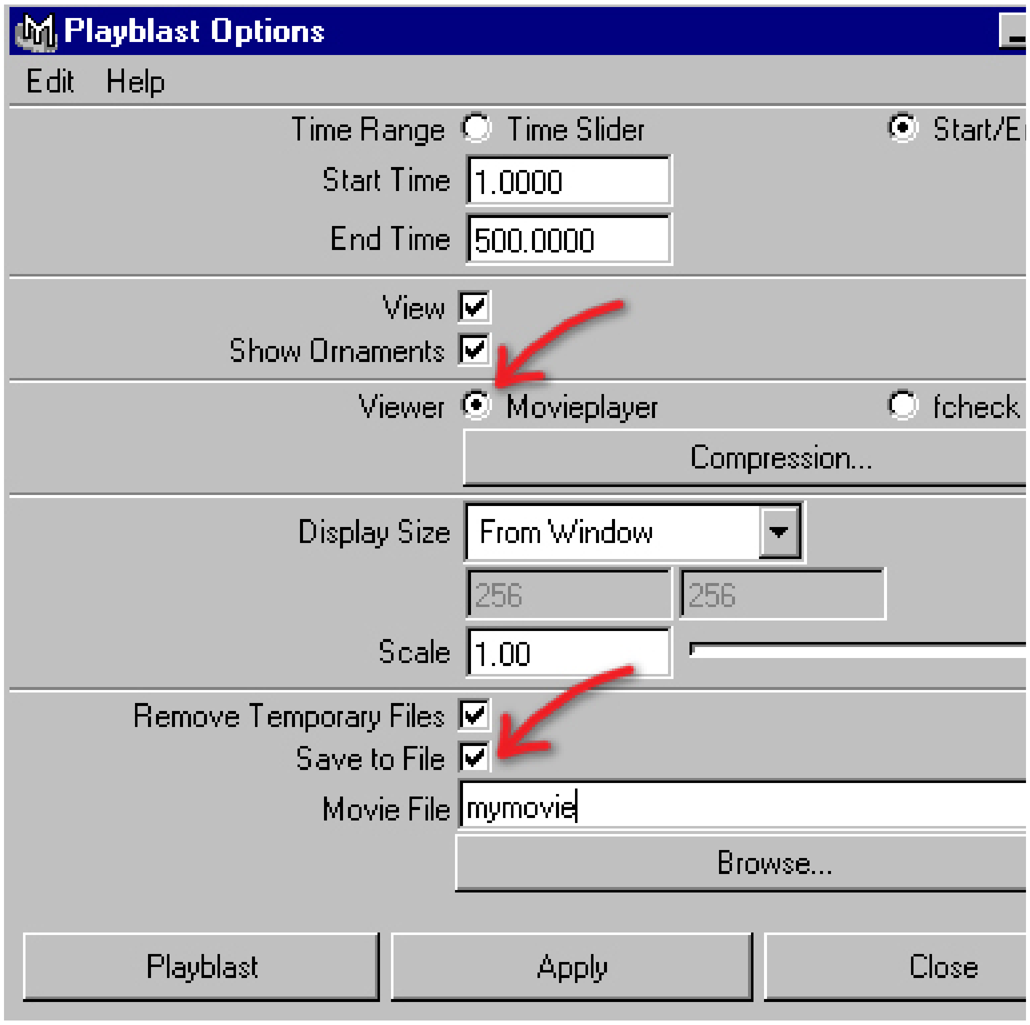
11. Create a movie file

To create a simple movie file of your own scene, you can use Maya's "playblast" function. This creates a hardware-rendered movie of your current window. You may want to un-hide any hidden objects, so that the truck and boxes are once again visible. To do this, use the Hypergraph window to select the nodes of the hidden objects. Then use **>Display>Show>Show Selected**.

Click on the Perspective window to make it active. Use **>Window >Playblast []**



In the window that opens, select Movieplayer next to Viewer. Check Save to File. Type in a name for the movie file. If you want to render the entire animation, select *Start/End*. Start Time should be 1, End Time should be the last frame of the animation. Hit Playblast.



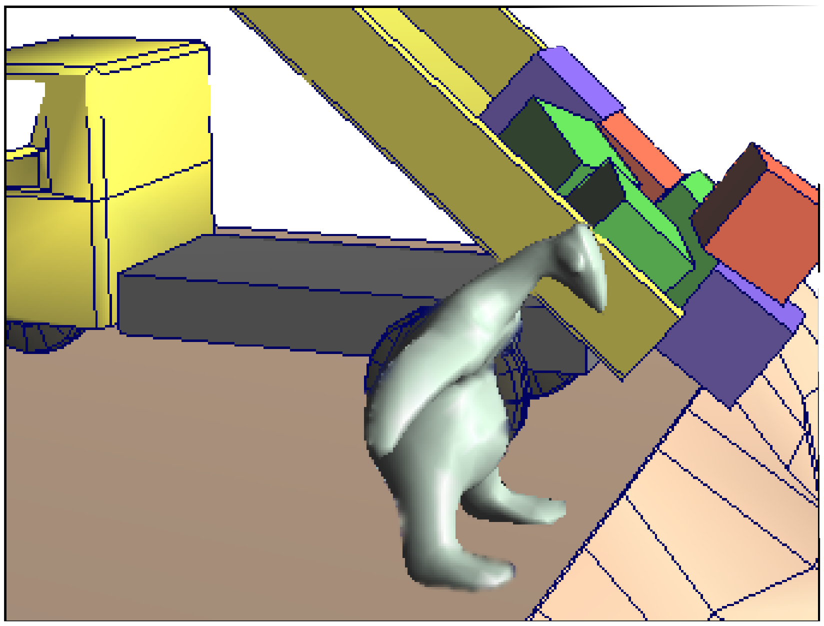
Time Range ○ Time Slider ● Start/E
Start Time 1.0000
End Time 500.0000
View ☑
Show Ornaments ☑
Viewer ● Movieplayer ○ fcheck
Display Size From Window
256 256
Scale 1.00
Remove Temporary Files ☑
Save to File ☑
Movie File mymovie

The movie file will open and play automatically as soon as it is finished. The saved .avi file is stored in your project's *images* folder

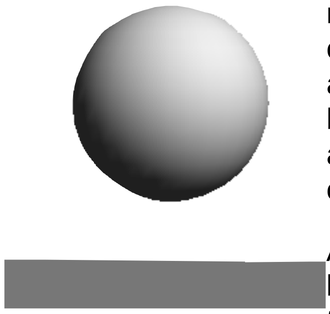
To see an example of a finished simulation for this exercise, use
>File >Import
Browse to find the *Course23/Demos* folder. Inside that folder, go to the *images* folder. Then, open the *Exercise3.mov* file.

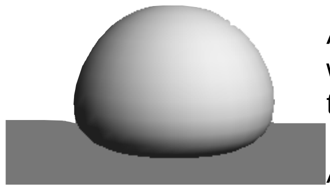
Lecture 2

Soft-Body Motion Dynamics



Soft Body Motion Dynamics





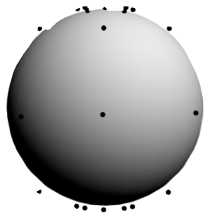
We have seen that motion dynamics calculations can be applied to a **rigid body**—that is, an object whose shape does not deform as it moves about. When motion dynamics calculations are applied to an object whose shape *does* deform as it moves about, such an object is frequently called a **soft body.** Soft body dynamics involve all the same parameters as rigid body dynamics, including forces such as gravity, and collisions with obstacle objects

An example of soft body dynamics would be a soft, rubber beach ball falling, bouncing off a table, and deforming as it bounces. In this case, the forces acting on the ball are gravity and the forces generated by the collision with the table.

Another example of soft body dynamics is a flag waving in the wind. Here, the forces might be wind and gravity, the combination of which cause the fabric of the flag to ripple in the breeze.

A third example of soft body dynamics might be a lock of hair on a character's head: as the character moves up and down in his walk, the velocity of his walking would generate up and down forces on the lock of hair, causing it to bounce up and down.
We have seen that when a rigid body moves in a simulation, the motion dynamics calculations act upon the object as a whole—the entire cube moves to the left or right; the entire cube bounces off the floor; etc. By contrast, when a soft body moves in a simulation, the motion dynamics calculations act upon the individual **surface points** of the object. If all of the object's surface points exactly the same amount at exactly the same time, there is no deformation of the object's shape. But if some of the surface points move more or less than others, the object's shape deforms.

In some software packages, the calculations of a motion dynamics simulation are done directly to the surface points of the soft-body object. Maya's implementaion is a bit different, however. When you create a soft body in Maya, the software creates a **particle system** and attaches it to your object. This particle system has exactly one particle for each of the object's surface points, and each surface point is associated with a specific particle. When a particle moves, its associated surface point moves. And when the surface points move, the surface deforms.

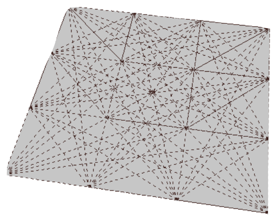


Maya's particle-system based implementation takes advantage of the fact that Maya already has very sophisticated and comprehensive software for the motion dynamics simulation of particle systems. By using particle systems to control the surface points of a soft body, Maya avoids the necessity of additional program code.

In the real world, a soft object has a coherent shape. It may deform, but it has a **structure** which results in it maintaining its basic configuration. For example, a rubber beach ball may squash when it hits something, but it quickly returns to its original, spherical shape. A flag waving in the wind, however much it ripples, retains its basic rectangular structure.

A motion dynamics soft body must also somehow retain its basic structure. The soft body deforms because its particles/surface points move around, but these points cannot move around without any constraints, in absolutely any direction or any amount. If they did so, the object would pull apart.

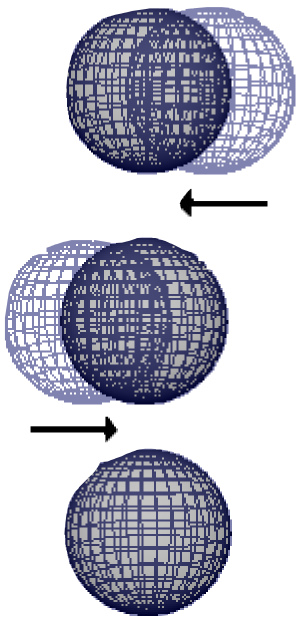
There are two approaches in motion dynamics to maintaining the structure of the soft-body object. The first involves the use of "**springs**". With this approach, the user defines a system of spring-like forces which connect all the points of the object together, as if each pair of adjacent points were connected by a rubber band.



An important parameter in any spring system is **stiffness**. This controls how much push-pull is permitted between adjacent points. To continue the rubber band analogy, it controls how flexible the rubber bands are. A stiffness of 100% means that there is no flexibility at all. In other words, the object cannot deform at all. If such an object collides with an obstacle, it behaves exactly as if it were a rigid body. On the other hand, a stiffness of 0% means that the points of the object can go anywhere. This is the same, in other words, as having no springs at all.

Another important parameter of springs is **damping**. This controls how quickly adjacent points return to their original positions —how quickly the rubber band snaps back. If damping is 0%, the points never stop moving—they bounce and jiggle forever. If damping is 100%, they stop moving immediately.

Both stiffness and damping can be adjusted either globally, for the entire spring system, or on a per-spring basis.



A second approach to maintaining the structure of a soft-body object is to use a "**goal**" object. With this approach, the original, non-soft object is considered the goal object for the soft-body object. As the surface points of the soft body move around and deform, each point is constantly trying to return to the position of its corresponding point on the goal object.

For example, imagine that a soft-body sphere has as its goal object the original configuration of the sphere before it became a soft body. If the goal sphere is moved suddenly to the left, the soft-body sphere will try to catch up to it. More specifically, each point of the soft-body sphere will try to catch up to each point of the goal sphere.

An important parameter when dealing with goal objects is the **weight** factor. This determines how closely the soft-body object will try to adhere to the goal object. A weight factor of 100% means that the soft object will adhere exactly to the goal at all times. In other words, they will be indistinguishable. A weight of 0% means that the soft object will not adhere at all to the goal. That is, it will not follow the goal at all.

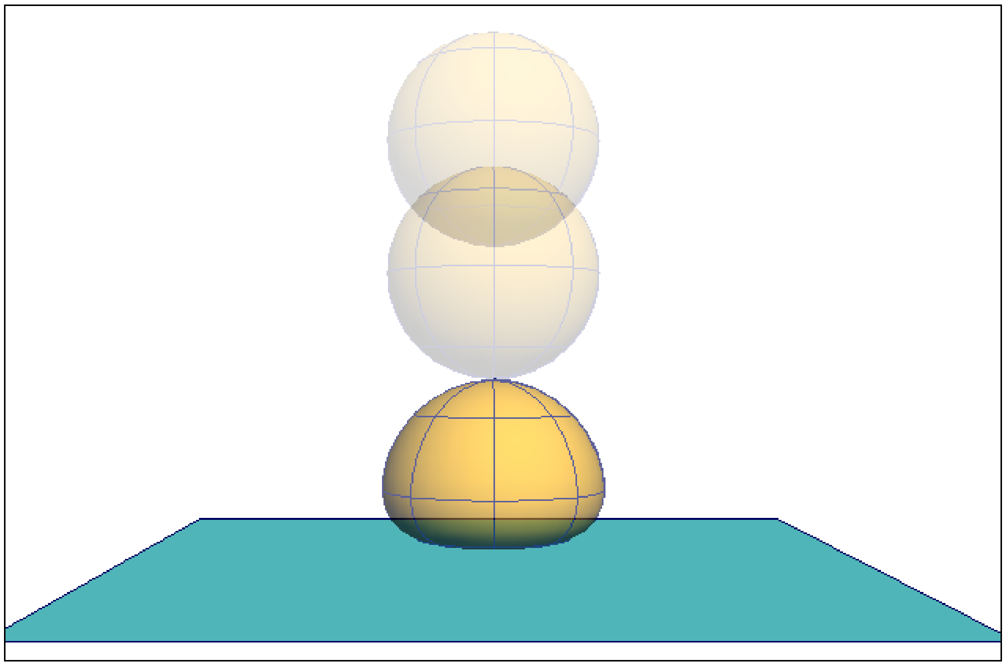
As with springs, parameters for goal objects can be adjusted either globally, to effect the entire object equally, or locally, to affect individual points individually. For example, for a soft-body lock of hair, the points near the base of the skull may have a weight of 100%, so that they do not move at all. The points at the tip of the hair might have a weight of 50%, so that they bounce about a lot. Intermediate points would have intermediate weight values.

As with any motion dynamics, soft-body objects can be animated using only motion dynamics calculations, or using some combination of keyframing and motion dynamics. To continue the lock of hair example, if the character is keyframed to walk along the street, his keyframed up-and-down movement will generate velocities which can be inherited by his soft-body hair. The result would be that as he walks and his body goes up and down, his hair would bounce up and down in response to his walking motion.

Exercise 4

Soft Bodies & Springs

In this exercise, we will make a soft-body ball deform as it bounces off a floor. We will control its deformation by defining a spring system for it. Once we have a simulation we like, we will convert the animation of the ball to permanent, keyframed information by "caching" the data.



1. Open the Starter Scene

Retrieve the starter Scene file for this exercise, using
>File >Open Scene

Remember that you must first locate the folder where the demo scene files are stored. Browse to find the folder *Course23/Demos.*

Inside that folder, go to the *scenes* sub-folder and open the scene file called *Exercise4-Start.*

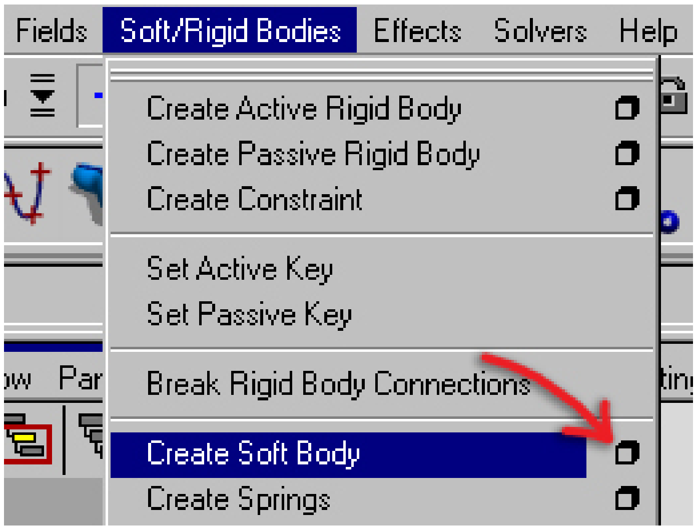
2. A Soft-Body Ball

In Exercises 1 through 3, all of our motion dynamics simulations were done with rigid bodies—that is, objects whose sh ape does not change. A motion dynamics object whose shape *does* change is called a "soft" body.

We will begin by making our ball a soft-body ball.

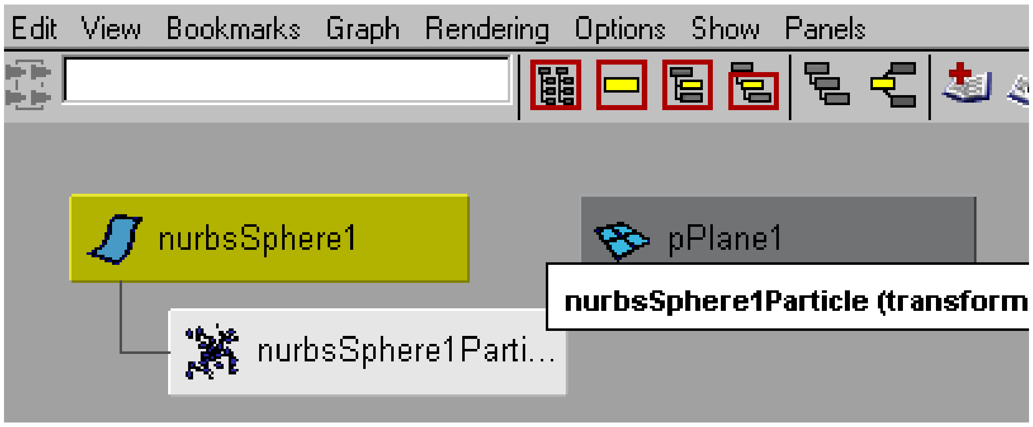
Make sure you are in the *Dynamics* module of Maya. You can use the F4 key to get to that module.

Select the ball by dragging your cursor over it. Then use
>Soft/Rigid Bodies >Create Soft Body []



Inside the window that opens, use **>Edit >Reset Settings**, to set all parameters to the default values. Click Create.

Look at the Hypergraph window. The ball should now have under it a particle-system node as a child within its hierarchy.


Edit View Bookmarks Graph Rendering Options Show Panels
nurbsSphere1
pPlane1
nurbsSphere1Particle (transform
nurbsSphere1Parti...

With the sphere still selected, use
>Fields > Gravity. You can use the default settings for gravity.

Remember that if the ball was selected when you created the gravity field, gravity will automatically be attached to the sphere. To confirm that it is, use
>Window >Relationship Editors >Dynamic Relationships

In the Dynamic Relationships window, select the ball on the left side of the window. On the right side of the Dynamic Relationships window, the gravity field should be highlighted . If it is not, click it to highlight it, thereby attaching it to the sphere.

Hit the Play button on the animation playback menu to start the simulation. The soft ball should drop under gravity.

Save your scene by using
>File >Save Scene As
Make sure you first browse to select the *Course23/Practice* folder, and its *scenes* subfolder Then type in a filename and save your file.

3. The Floor as Collision Obstacle

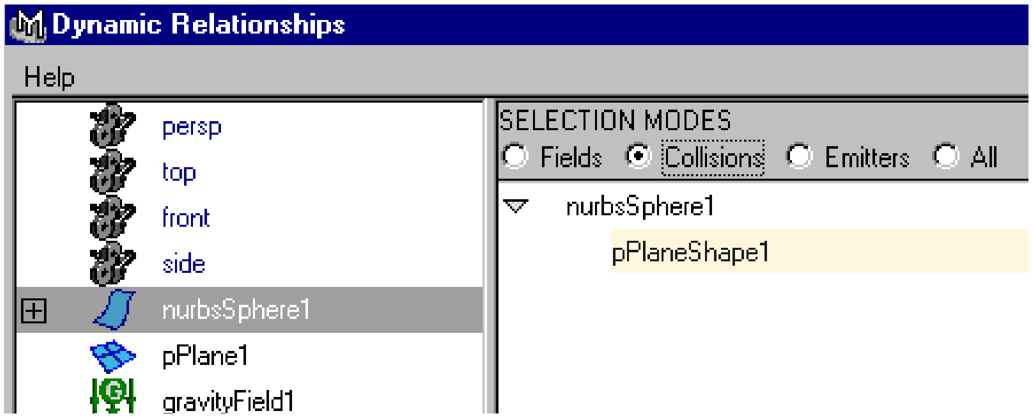
In Maya, obstacle objects are defined differently for soft bodies than they are for rigid bodies. In Maya soft-body dynamics, it is the particles of the soft body which collide with other objects, not the original objects themselves.

Begin by selecting the soft body's particle system. One way to do this is to click on one of the particle dots in any of the modeling windows. (If you can't see the particle dots, use the menu at the top of the Perspective window. Use **>Show>Dynamics**.) Another way is to click on the ball's particle node in the Hypergraph window.

With the particle system node selected, shift-select the floor so that both the particle node and the floor are selected. Then use
>Particles >Make Collide

This defines the floor as an obstacle for that particle system.

Confirm this with
>Window >Relationship Editors >Dynamic Relationships
Click on the ball in the left side of that window.
On the right side, click on Collisions at the top. The floor should be highlighted. If it is not, click on it to highlight it, thereby connecting it to the soft-body ball.



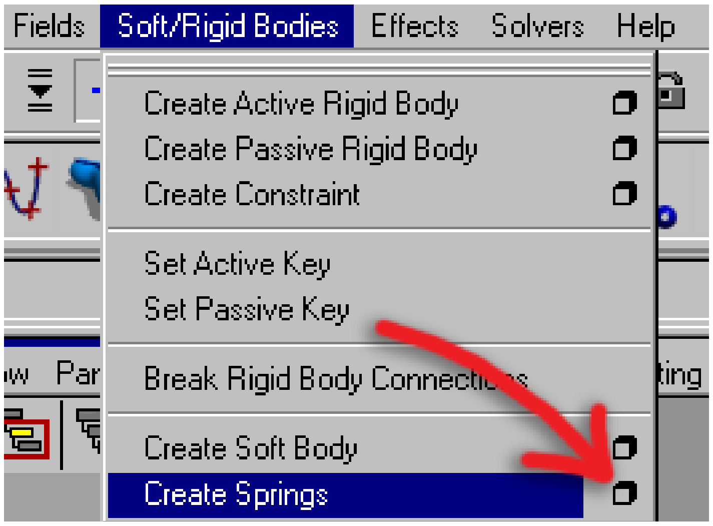
Play the simulation by clicking on the animation Play button. The soft ball should drop, collide with the floor, and deform when it collides.

But it deforms way too much! We will correct that in a moment, but first, save your scene with
>File >Save Scene As

4. Add Springs to the Ball

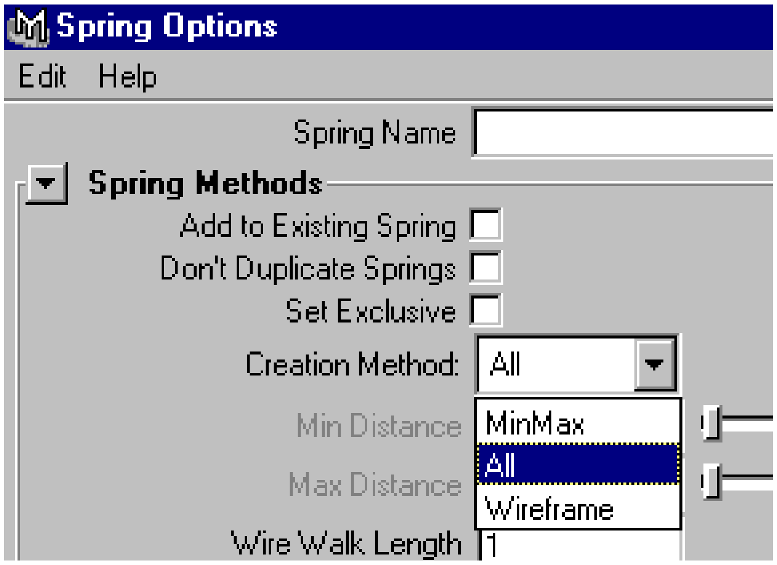
In order to control the deformations of the soft-body ball, we will create a set of "springs".

Select the ball. With the ball selected, use
>Soft/Rigid Bodies >Create Springs []



Within this window, use **>Edit >Reset Settings**, to get the default settings.

Still inside the *Create Springs* window, next to Creation Method, select All. This step is important, because it will cause all particles of the soft body to be connected to each other.
Select Create to create the spring system for the soft-body ball.



Now play the simulation again by hitting the animation Play button.

The ball should deform much less when it collides with the floor. The springs are holding the particles of the soft body together.

5. Playback Speed-ups

Adding springs to the soft body slowed down your animation playback considerably. This is partly because of all the additional calculations. It is also because your screen image is more complex now.

You can speed up playback by simplifying your screen image.

First, at the top of the Perspective window, use
>Show >Dynamics
This will turn off the display of all the springs and other dynamics icons. Play the simulation now. It should play more quickly.

If necessary, you can also simplify your display further by making the Perspective window full-screen. You can do this by placing your cursor over the Perspective window, then hitting the space-bar key on your keyboard. The Perspective window should pop to full-screen. (To go back to multiple windows, hit space-bar again.)

Save your scene with
>File >Save Scene As

6. Adjust Spring Parameters

To fine-tune the way the springs exert their control over the ball's deformation, we will adjust some of the spring parameters.

Select the *springs* node. The easiest way to do this is by clicking on the *springs* icon in the Hypergraph window. (If you turned off the display of dynamics in the modeling window, this will be the *only* way you can select it.)

With the springs node selected, open its Attribute Editor. You can do this either with Ctrl-a, or with **>Window >Attribute Editor**. Select the *springShape* tab.

The most significant parameters for springs are usually *stiffness* and *damping*.

Damping controls how quickly the springs stop stretching. Inside the springs' Attribute Editor, change the damping number to 0.0. Play the animation. The ball should continue deforming forever.

Change damping back to the default value of 0.2.

Stiffness controls how much the springs stretch. Still inside the Attribute Editor, change the stiffness number to 3.0 Play the animation. The ball should deform less than before.

Careful! If the stiffness and damping numbers are too large or too small, the simulation calculations fail, and the soft body object "explodes"!

Save your scene with
>File >Save Scene As

7. Cache your Particle Animation

Concept

Just as with rigid-body dynamics, we want to fix our simulation animation into a permanent form. With rigid bodies, Maya's technique for doing this is "baking" the simulation.

For soft bodies, the process of fixing the animation into a permanent form is different. This is because for soft bodies, data needs to be saved, not merely for every object, but for every *particle* of every object.

Maya's technique for doing this is "particle caching".

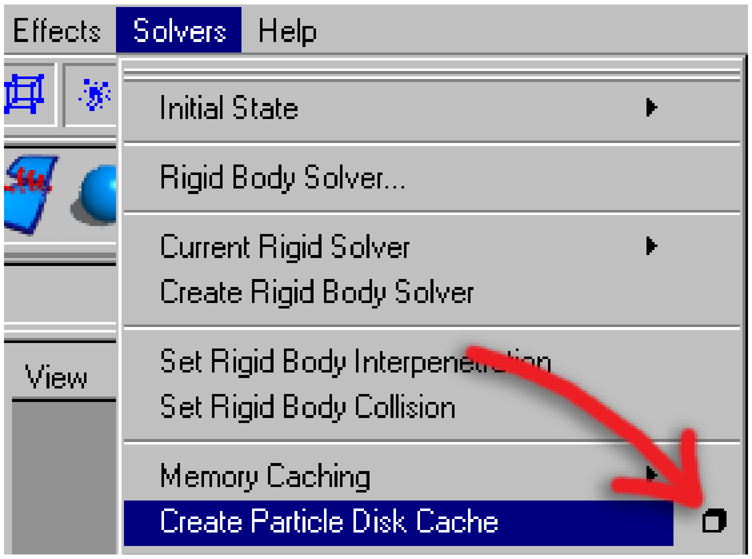
"Cache" means to store. Maya's process is to store data for all the particles of your soft body for every frame of the animation. This data can be stored either temporarily in RAM, or permanently on your hard drive. We will store the data on the hard-drive.

Create the cache files

The procedure is in three parts.

A) Save your scene file with **>File >Save Scene As**. Give your file a new name so that you do not overwrite your previous version. Saving the scene file first is necessary so that Maya will know what filenames to give to the cache files it is about to create.

B) Select the soft-body ball. With the ball selected, use
>Solvers >Create Particle Disk Cache []



Inside that window, use
>Edit >Reset Settings, to get the default settings. The data files will be stored in a particular subfolder, or directory, which Maya calls the "cache directory." Next to *Cache Directory*, type in a name for this folder, usually the same name as your scene file. Then hit Create.

You will see the frame counter advance across the time slider as the data is calculated. The animation will not update in the modeling windows, however.

Once the cache data has been created, Maya will read from it every time you play the animation, rather than calculate the motion dynamics at every frame, as it had been doing. Try dragging the frame counter with your cursor. You should see that the animation follows your frame counter perfectly.

C) The third step is to re-save your scene file under the same name. This is so that Maya will know the next time you open your scene file, that the cache files are there. Use **>File >Save Scene**.

Using the cache files

Once the cache files are saved onto your hard drive, you do not need the motion dynamics calculations at all. In the Hypergraph window, select both the *gravity* and *springs* icons. Delete them.

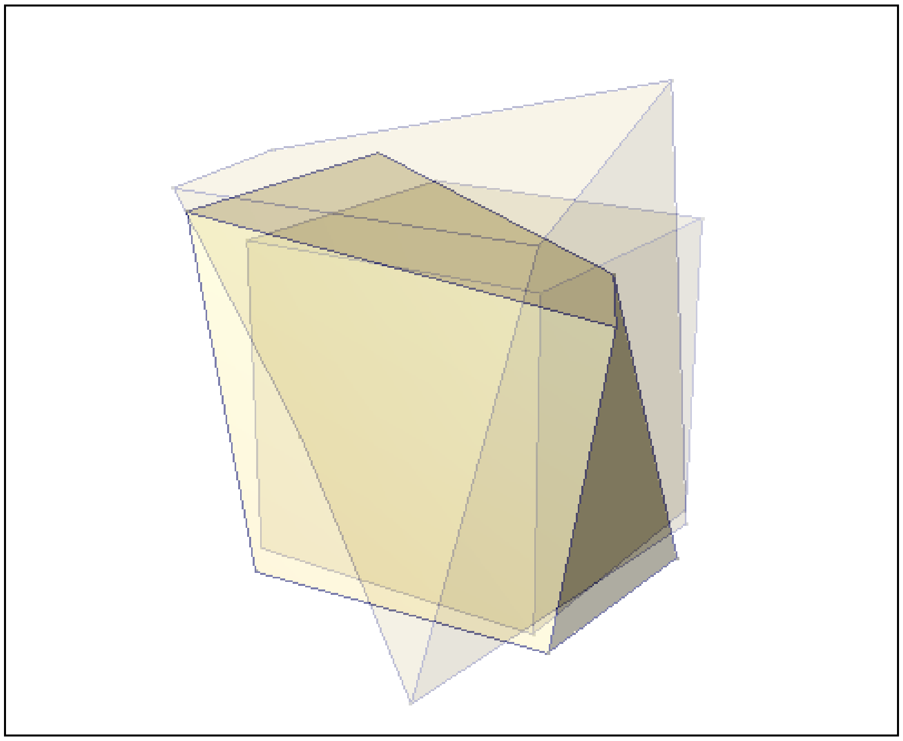
Play the animation. Even though gravity and springs have been deleted, the animation plays properly, because the motion dynamics calculations are no longer been done.

Note that you can *not* delete the particle node, however. This is because the cache information is telling Maya how to move those particles. If you delete the particle node, the ball will no longer have any animation at all.

Exercise 5

Soft Bodies & Goal Objects

In this exercise, we will make a soft-body cube, and control its deformation by a goal-object. First, the deformations will be uniform for the entire cube. Then we will modify the deformations by adjusting goal weights, to give an irregular deformation to different parts of the cube.



1. Get the starter scene

Retrieve this exercise's starter scene with
>File >Open Scene

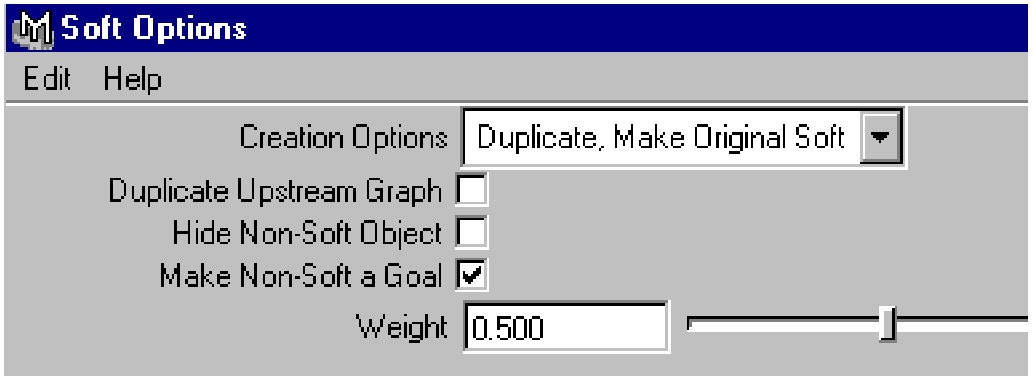
Remember that you must first locate the folder where the demo scene files are stored. Browse to find the folder *Course23/Demos.*

Inside that folder, go to the *scenes* sub-folder and open the scene file called *Exercise5-Start.*

2. Make the Soft Body

Make sure you are in the Dynamics module by hitting the F4 key.

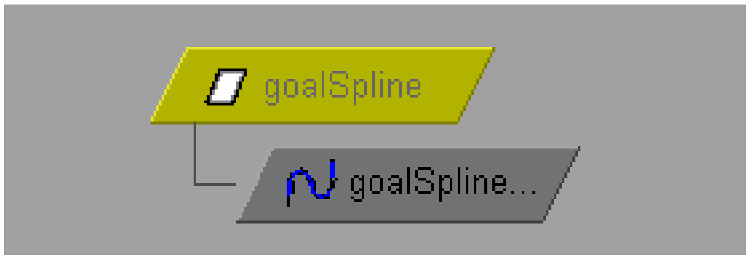
Select the cube by dragging your cursor over it. Use
>Soft/Rigid Bodies >Create Soft Body [].
Inside the window that opens, next to Creation Options, select Duplicate, Make Original Soft. Also check on Make Non-Soft a Goal. Leave the Weight at the default 0.5. Click Create.


Soft Options

Edit Help

Creation Options: Duplicate, Make Original Soft
Duplicate Upstream Graph ☐
Hide Non-Soft Object ☐
Make Non-Soft a Goal ☑
Weight 0.500

Look in the Hypergraph window, which should be in your upper-left panel. (If it is not, you can use **>Window >Hypergraph**.) You should see that there are now two objects—the original cube has become a soft-body object, and has a particle system node under it. The duplicate cube is now the goal object.



Rename the goal object to "goal". You can do this in the Hypergraph window. Place your cursor over the goal cube (it should be called something like "*copyOfpCube1*"). Hold down the right mouse button, and drag down to "Rename"; type in the new name, "*goal*".

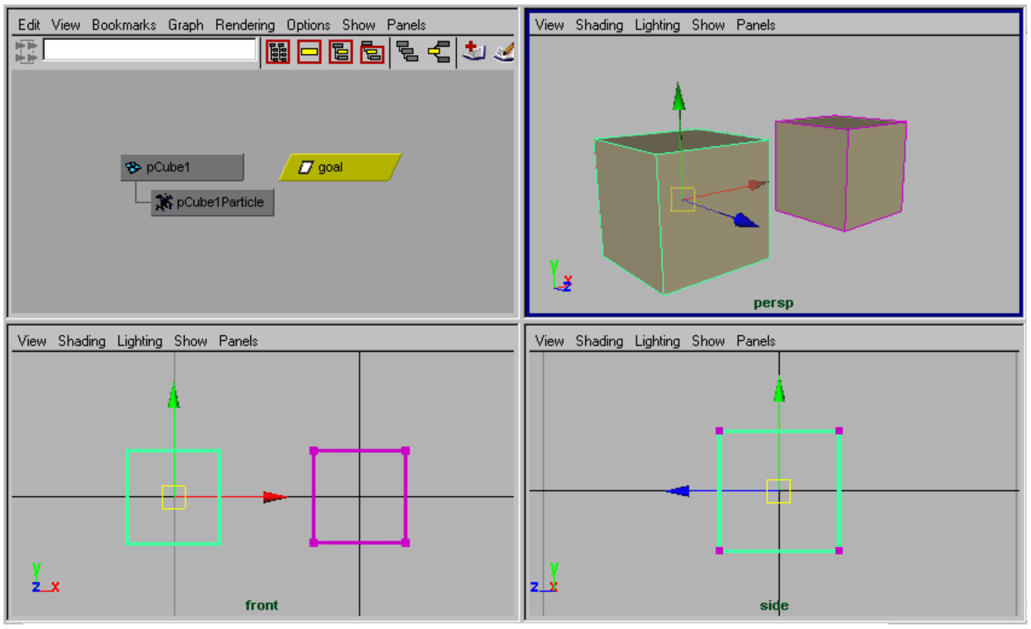
3. Keyframe the Goal Object

Select the goal cube. It will be easiest to do this in the Hypergraph window, since both the soft cube and the goal cube are in exactly the same place.

Move the frame counter at the bottom of the screen to frame 1.

With the goal cube selected, hit the Shift-w key (upper-case w). This will keyframe the Move, or Translation, values of the goal cube at frame 1. Notice in the Hypergraph window that the goal cube's icon now is a slanted parallelogram, to indicate that it has animation.

Now drag the frame counter to frame 5. Hit the lower-case w key to go into Move mode. Drag the goal cube to the left—that is, negatively along the X axis—by dragging the red axis of the Move icon. Position the goal cube at about -10.0 in X. Now hit the Shift-w key to save a keyframe for the goal cube at that position at frame 5. Drag the frame counter back and forth in the time slider to make sure your goal object is animated.


Edit View Bookmarks Graph Rendering Options Show Panels
pCube1
goal
pCube1Particle
View Shading Lighting Show Panels
persp
front
side

Play the animation by hitting the Play button on the animation playback menu.

The goal cube moves abruptly from its initial position at frame 1 to its position at frame 5. The soft cube tries to stay attached to the goal cube. The result is that the soft cube overshoots, then recoils, then bounces back a few times, before finally coming to rest in the same position as the goal cube.

To see the movement of the soft cube more easily, we can hide the goal cube. In the Hypergraph window, select the goal cube by clicking on it. With the goal cube selected, use the Ctrl-h key combination. (Alternatively, you could use **>DIsplay >Hide >Hide Selection**)

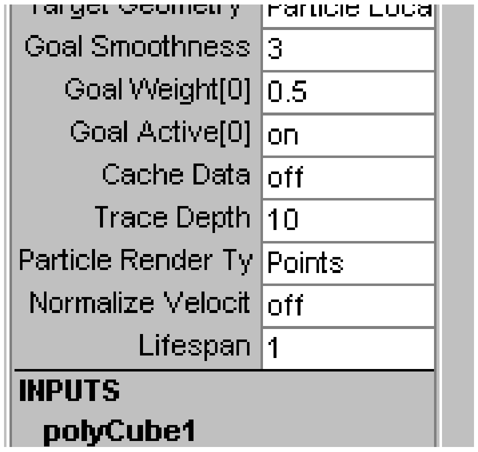
Save your scene with
>File >Save Scene As
Remember to put your scene in the folder called *Course23/Practice.*

4. Adjust Global Goal Weight

To control how closely the soft object sticks to the goal object, we can adjust the Goal Weight parameter of the soft object. This parameter is found under the soft object's particle system.

Select the soft cube's particle system. The easiest way to do this is by clicking on its icon in the Hypergraph window.

With the particle node selected, you can now adjust the Goal Weight parameter. One way to do this is by scrolling through the parameters that show up now in the **Channel Box**, on the right side of your screen. (Alternatively, you could go to the Goal Weights and Objects section of the particle node's Attribute Editor.)



Target Geometry	Particle Loca
Goal Smoothness	3
Goal Weight[0]	0.5
Goal Active[0]	on
Cache Data	off
Trace Depth	10
Particle Render Ty	Points
Normalize Velocit	off
Lifespan	1

INPUTS
polyCube1

Using the Channel Box, change the Goal Weight of the particle node to 0.8. This causes the soft cube to stay much more closely attached to the goal cube. The soft cube now snaps very quickly to the goal cube, and does so with very little bouncing.

Try changing the Goal Weight to some other numbers between 0 and 1. A Goal Weight of 1.0 causes the soft cube to stay 100% perfectly attached—that is, there is no difference at all between the soft and the goal objects. A Goal Weight of 0.0 causes the soft cube to not attach at all.

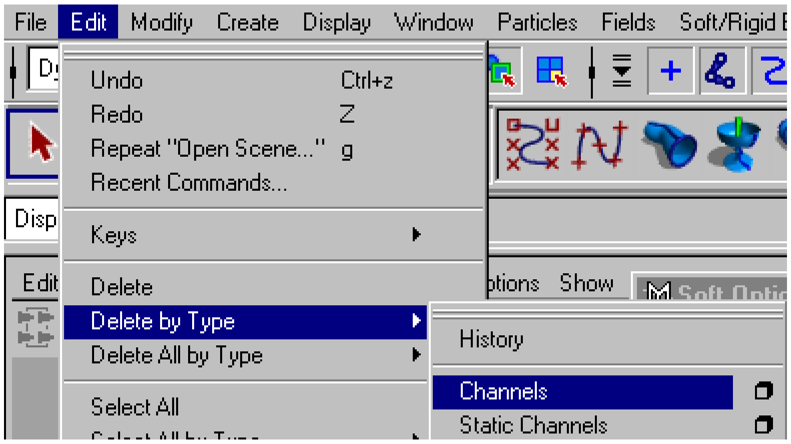
Save your scene with
>File >Save Scene As

5. Add a Turbulence Force

We will now delete the animation we have so far and create a new animation with a turbulence field.

Delete the keyframed animation

Begin by deleting the animation on the goal cube. In the Hypergraph window, select the goal cube icon. Now use
>Edit >Delete by Type >Channels

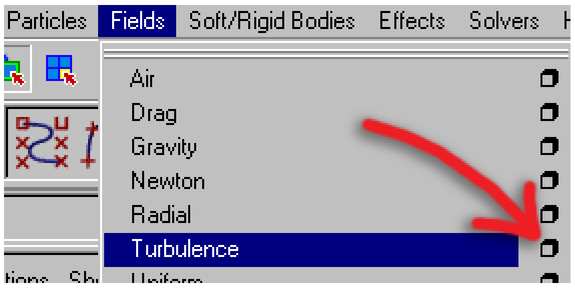


"Channel" is a word used in Maya to indicate a set of animation information. Deleting the channel deletes the animation. The goal cube's icon in the Hypergraph should now revert to being a rectangle, indicating that it has no animation.

Play the animation to make sure this is the case. You should have no animation.

Create a turbulence force

Select the soft cube. With the soft cube selected, use
>Fields >Turbulence []



In the option window that opens up, set *Magnitude* to 300, to make the turbulence stronger. Set *Attenuation* to 0.0, to prevent it from dying out with distance.

Play the animation. The cube should jiggle about. Each of its particles is being pushed around randomly by the turbulence field. At the same time, the particles are trying to retain the shape of the goal cube. (What a quandry for the poor particles!)

Try adjusting some of the parameters of the turbulence field. First, select the turbulence icon in the Hypergraph window. You can now find its parameters either in the Channel Box on the right side of your screen, or in the Attribute Editor (>Window >Attribute Editor). Try changing both the magnitude and attenuation.

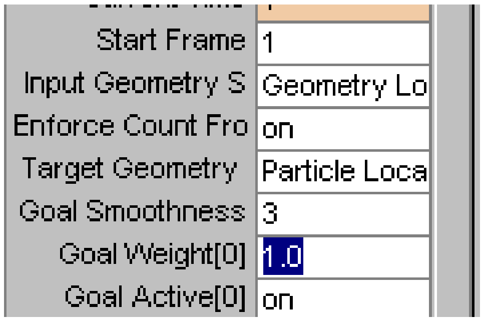
Select the particle system node in the Hypergraph. In the Channel Box, change the Goal Weight parameter to a lower number—for example, 0.3. Play the animation. Set the Goal Weight back to 0.5

Save your scene with
>File >Save Scene As

6. Adjust Per-particle Goal Weight

Concept

So far, we have been adjusting the soft cube's goal weight on a global basis. The result has been that whatever happens to the soft cube, it happens uniformly. That is, all the particles of the soft cube react the



Start Frame	1
Input Geometry S	Geometry Lo
Enforce Count Fro	on
Target Geometry	Particle Loca
Goal Smoothness	3
Goal Weight[0]	1.0
Goal Active[0]	on

same amount to whatever forces are acting on them

It is possible to make different parts of the soft object react more or less than other parts of the same object. To do this, you define indivdual goal weights to individual particles. Maya refers to this as "per-particle goal weights".

The final goal weight of an individual particle is a result of its own per-particle goal weight multiplied by the global goal weight of the entire particle system. For example, if the global goal weight is 0.5, and we then assign a per-particle goal weight of 0.5 to one of the particles, the final goal weight of that particle will be 0.25 (0.5 * 0.5 = 0.25).

Reset the Global Goal Weight

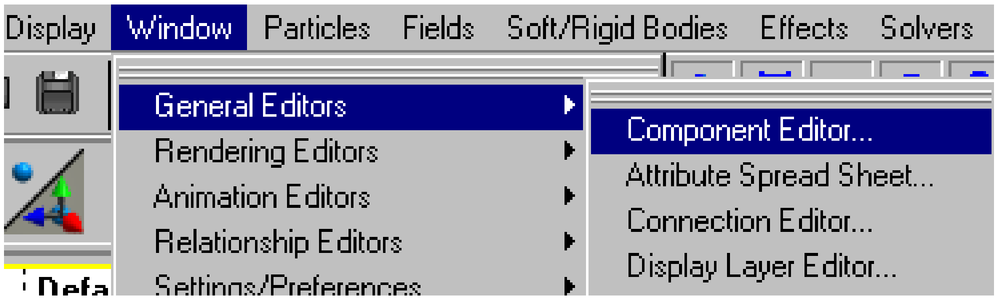
Because of the multiplication process just described, it is helpful to assign a value of 1.0 to the global goal weight of the particle system before beginning to assign any per-particle goal weights.

So, select the soft cube's particle system in the Hypergraph window. In the Channel Box, set Goal Weight to 1.0

The Component Editor

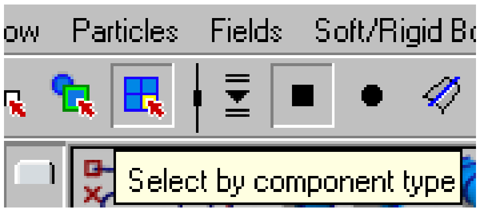
Maya provides two ways of adjusting the per-particle goal weights of a soft object. The one we will use here allows you to adjust the goal weights numerically. This is done with Maya's Component Editor.

First, select the soft cube's particle node. Now use
>Window >General Editors >Component Editor

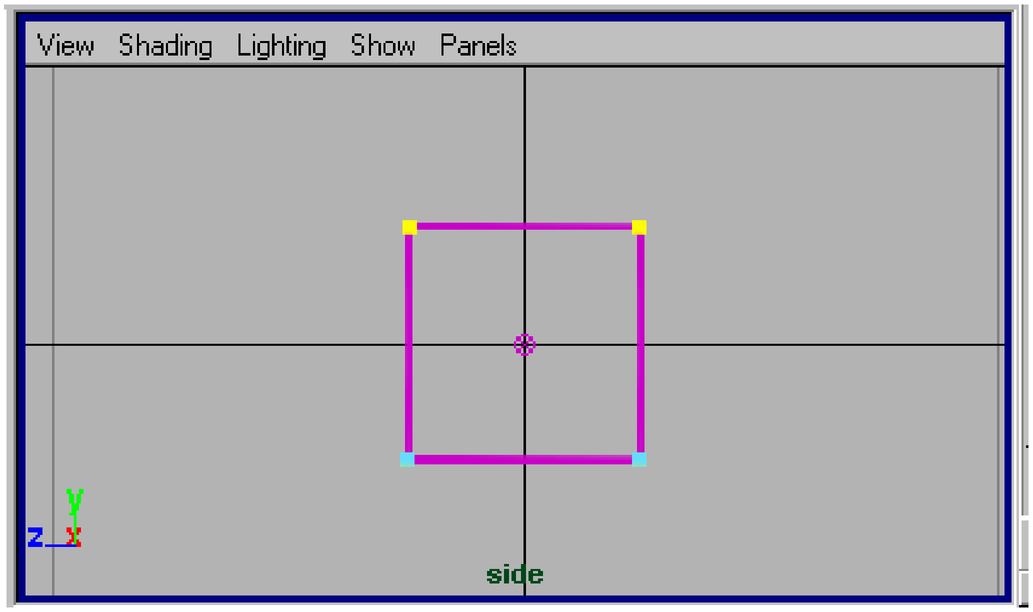


The Component Editor allows you to change a variety of parameters for individual components.

Now we must select the particles we want to operate on. At the top of your screen, click the little Component Mode selection button, to allow you to select components, rather than objects.



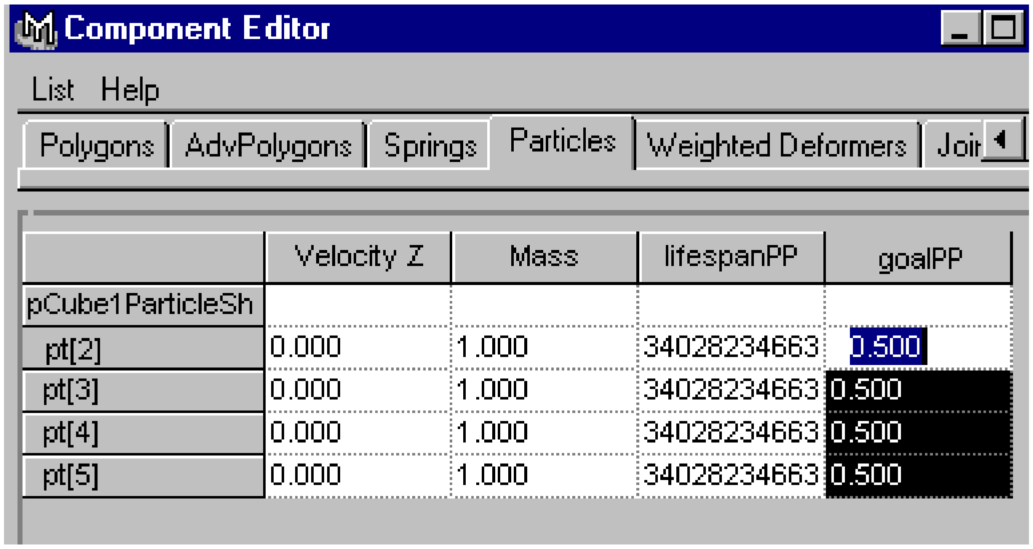
Now, in the Front or Side window, drag your cursor over the particles at the top of the cube to select them.



Now that the particles are selected, go back to Component Editor window. In that window, click the *Particles* tab at the top of the window to display parameters for particles. (You may need to resize the window to see the Particles tab). As soon as the particles are selected, the Component Editor will display the values for all of their parameters.

The parameter we are interested in is on the far right of the Component Editor window. It is called *goalPP* (goal per particle). You should see that all of these have a value of 1.0 (because we set the global goal weight to 1.0).

Click and drag over all the 1.000 values under the *goalPP* column. With these selected, type in 0.5. As soon as you hit Enter, the *goalPP* value for all four of the particles you selected changes to 0.5.



	Velocity Z	Mass	lifespanPP	goalPP
pCube1ParticleSh				
pt[2]	0.000	1.000	34028234663	0.500
pt[3]	0.000	1.000	34028234663	0.500
pt[4]	0.000	1.000	34028234663	0.500
pt[5]	0.000	1.000	34028234663	0.500

Play the animation. The soft cube now deforms only at the top. The bottom of the cube does not deform at all, because all of the particles at the bottom have a goal weight of 1.0

Try selecting just one of the particles. (It will be easiest to do this in the Perspective window.) Use the Component Editor to change its *goalPP* number to 0.3. Play the animation. That one particle should now jump around more than any of the others.

7. Cache your Particle Animation

Use the same techniques that we saw at the end of Exercise 4 to create cache files for the particles of your soft-body animation, in order to make it permanent.

In order to do this, remember that you must
A) save your scene file with **>File >Save Scene As**
B) select the soft-body cube, and use **>Solvers >Create Particle Disk Cache[]**
C) re-save your scene file with **>File >Save Scene As**

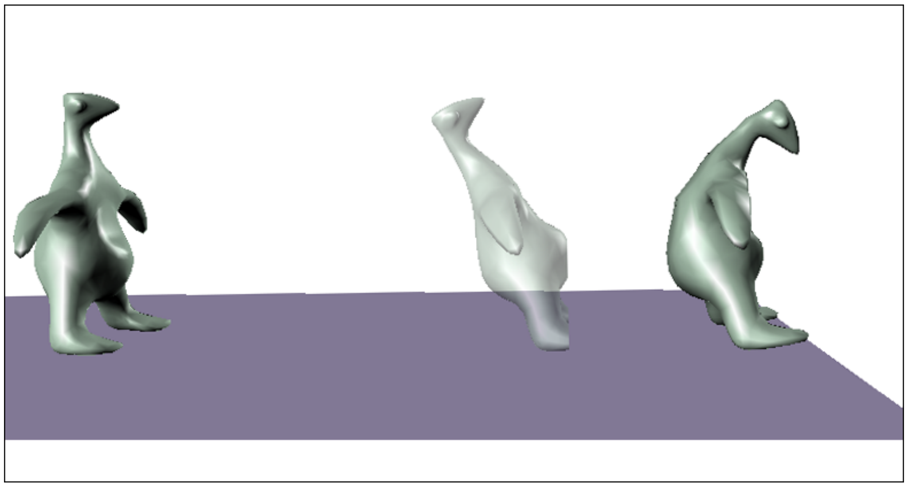
Once you have done all this, delete both the goal-object cube and the Turbulence field. Since the motion dynamics calculations are no longer needed, these dynamic elements no longer serve a function. After deleting these, play the animation to make sure your cached animation is ok.

Save your scene with **>File >Save Scene As**. Make sure to give it a new name, so that you do not overwrite the version which does have the motion dynamics in it.

Exercise 6

A Soft-Body Character

In this exercise, we will animate a character using soft-body dynamics. The character will slide across the floor and bounce and jiggle as a result of the soft-body calculations.



1. Get the starter scene

Retrieve this exercise's starter scene with **>File >Open Scene**.

Remember that you must first locate the folder where the demo scene files are stored. Browse to find the folder *Course23/Demos.*

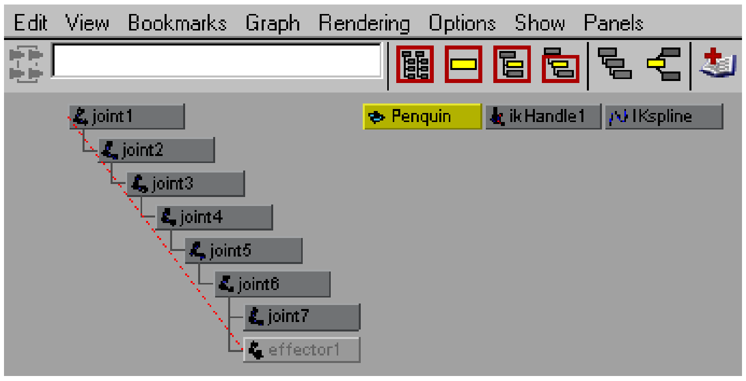
Inside that folder, go to the *scenes* sub-folder and open the scene file called *Exercise6-Start.*

2. Understand the Rigging

It is important first to understand how this character model will be controlled. The mechanisms for controlling the motion of a character are often referred to as the character "rigging". The rigging for this character has already been provided for you here in this file.

The skeleton

The model for our little character is a single polygonal object, called *Penquin* in your Hypergraph window. Inside that object is a simplified Inverse Kinematic (IK) skeleton. For our character, this skeleton is a simple series of joints going straight up from the floor to his (her?) head. In the Hypergraph window, this skeleton is indicated by a hierarchy of nodes called *joint1*, *joint2*, etc..



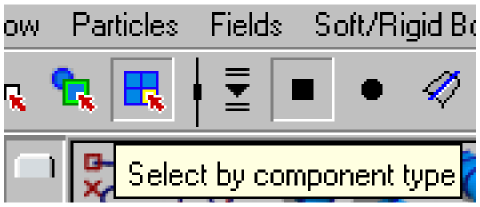
The geometry of the character model is "bound" to the IK skeleton. This means that if the skeleton moves or changes shape, it will cause the geometry of the Penquin to move or deform.

The IK spline

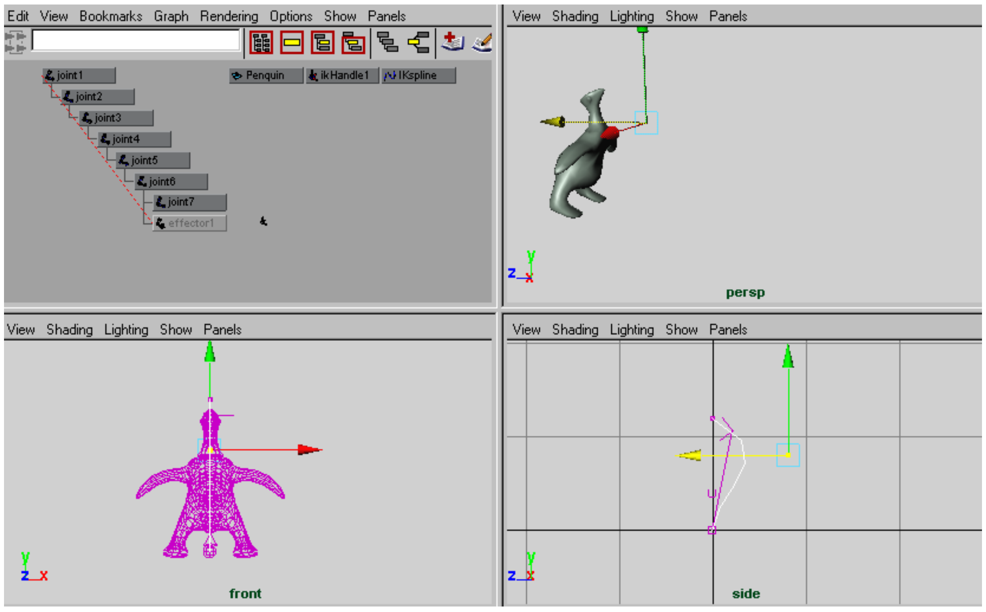
The most common way to manipulate an IK skeleton is to move its "effector". In Maya this is accomplished by creating an "IK handle" at the tip of the skeleton. This handle can then be moved to control the skeleton.

Our character has been rigged with a different technique, however. Here, we have created an "IK spline handle" to control the skeleton. This is a control created by a curve that runs up the length of the skeleton. If we change the shape of this IK spline curve, we change the shape of the skeleton - and this in turn changes the shape of the geometry.

Before proceeding, test out the use of this IK spline. The lower right window of your screen has been set up to display only the IK spline. Drag your cursor to select that IK spline. Now go into component selection mode by clicking the little icon at the top of your screen.

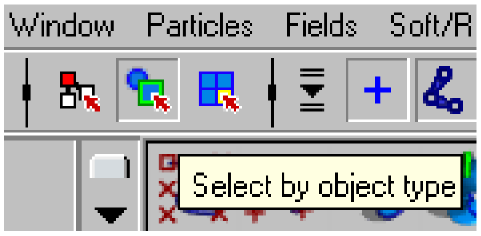


The control points of the IK spline should be displayed. Drag your cursor over one of them to select it. Hit the w key on your keyboard to go into Move mode. Click on one of the axes of the move icon and drag it. As the control point of the IK spline is moved, the skeleton deforms, which in turn causes the geometry to deform



Hit the z key to undo your move and put the IK spline back in its original position.

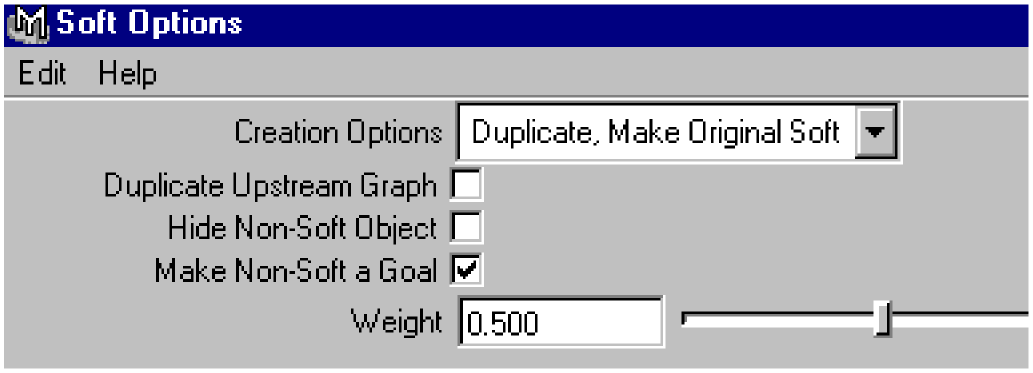
Click the Object Selection button at the top of your screen to back into object selection mode.



3. Make the IKspline a Soft Body

Just as we did in Exercise 5, we will make a soft body with a goal object. In this exercise, however, the soft body will be the IK spline curve.

Select the IK spline curve. The easiest way to do this is through the Hypergraph window. With the IK spline selected, use
>Soft/Rigid Bodies >Create Soft Body []
Inside the option window that opens, set *Creation Options* to Duplicate, Make Original Soft. Then turn on Make Non-Soft a Goal. Then hit Create.



Creation Options: Duplicate, Make Original Soft
Duplicate Upstream Graph ☐
Hide Non-Soft Object ☐
Make Non-Soft a Goal ☑
Weight 0.500

In the Hypergraph, notice that the IKspline node now has a particle node under it, indicating that it is now a soft body. We will rename the IKspline node. Place your cursor over the IKspline node, hold down the right mouse button, drag to Rename, and type in *softSpline* as the new name.

Notice also in the Hypergraph that there is now a copy of the IK spline.

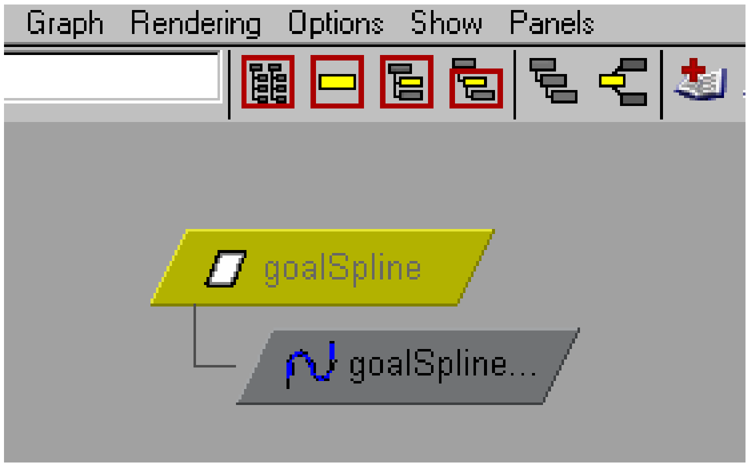
It should be called *copyOfIkspline*. Rename that node to *goalSpline*.

Save your scene by using
>File >Save Scene As
Remember that all your exercise files should be stored in the folder called *Course23/Practice/scenes*. If necessary, browse to find that folder.

4. Keyframe the Goal Spline

Now we will keyframe the goal spline. Wherever the goal spline goes, the soft spline will follow - wherever the soft spline goes, the skeleton will follow, and wherever the skeleton goes, the Penguin model will follow.

Select the goal spline. This will be easiest to do in the Hypergraph window. Drag the frame counter over to frame 1. Now, hit the Shift-w keys (upper-case W) , to save a translation keyframe for the goal spline at frame 1.



Drag the frame counter to 10. Hit the lower-case w key to go into Move mode. Drag the Move icon to move the goal spline negatively along the Z axis about 15 units—that is, to about 0, 0, -15. Hit the Shift-w keys to save that keyframe at that position.

Go back to frame 1. Hit the Play animation button. The Penguin jumps over to the right, and bounces back and forth for a while. This is because the soft spline is trying to catch up to the goal spline.

We can control the amount of bouncing by changing the goal weight of the soft spline's particle node. In the Hypergraph, select the particle node. In the Channel Box on the right side of your screen, change the goal weight from the default 0.5 to another number. Play the animation. The Penguin's bounce should be different.

Save your scene by using
>File >Save Scene As

5. Adjust the Goal Weights

We will now adjust the goal weights of individual particles, just as we did in Exercise 5.

Reset the Global Goal Weight

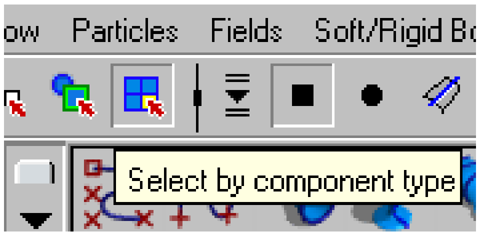
We will begin by setting the goal weight of the entire particle system to 1.0. In the Hypergraph window, select the particle system node of the soft spline. In the Channel Box, set Goal Weight to 1.0.

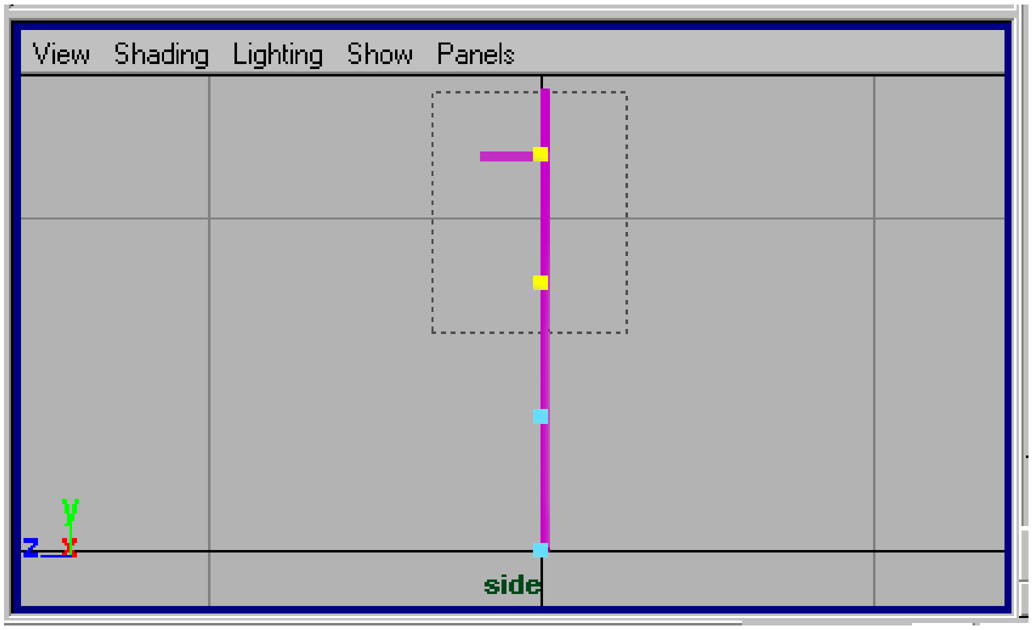
The Component Editor

Now we will use the Component Editor to change the goal weights of individual particles. Use
>Window >General Editors >Component Editor
Click the Particles tab inside that window.

Make sure soft spline's particle node is still selected.

Now we will select the particles we want to operate on. At the top of your screen, click the little Component Mode selection button.





In the Side window, where you see only the soft spline, drag your cursor over the two particles at the top of the spline to select them. They should be highlighted as yellow when they are selected.

As soon as the particles are selected, the Component Editor will display the values for all of their parameters.

On the far right of the Component Editor window, click on *goalPP* value of your selected particles. Type in 0.5.

Play the animation. The top portion of the Penguin bobs back and forth, because the particles at the top of the soft spline have goal weights of 0.5. The bottom portion of the Penguin does not bob back and forth at all, because the particles there have goal weights of 1.0.

Fine tune the animation by selecting other particles on the soft spline and changing their goal weights. You can also modify the animation by changing the keyframed positions of the goal spline.

Save your scene by using
>File >Save Scene As

6. Cache your Particle Animation

Use the same techniques that we saw at the end of Exercise 4 to create cache files for your the particles of your soft-body animation, in order to make it permanent.

In order to do this, remember that you must
A) save your scene file with **>File >Save Scene As**
B) select the soft-body cube, and use **>Solvers >Create Particle Disk Cache[]**
C) re-save your scene file with **>File >Save Scene As.** Make sure to give it a new name, so that you do not overwrite the version which does have the motion dynamics in it.

Once you have done all this, delete the goal spline. It is no longer needed, since motion dynamics calculations will no longer be calculated.

Play the animation to make sure your cached animation is ok.